Insufficient Gas Griefing

Description

Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract. If the sub-call fails, either the whole transaction is reverted, or execution is continued. In the case of a relayer contract, the user who executes the transaction, the 'forwarder', can effectively censor transactions by using just enough gas to execute the transaction, but not enough for the sub-call to succeed.

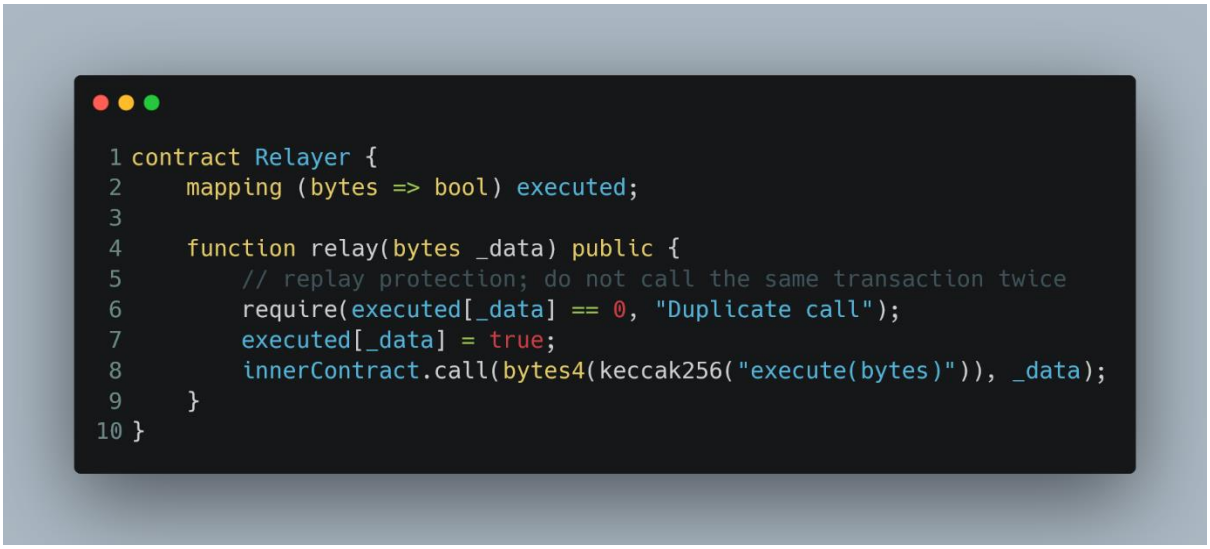
Remediation

There are two options to prevent insufficient gas griefing:

- Only allow trusted users to relay transactions.
- Require that the forwarder provides enough gas.

Example:

Code:



```
contract Relayer {
    mapping (bytes => bool) executed;

    function relay(bytes _data) public {
        // replay protection; do not call the same transaction twice
        require(executed[_data] == 0, "Duplicate call");
        executed[_data] = true;
        innerContract.call(bytes4(keccak256("execute(bytes)")), _data);
    }
}
```

Explanation:

Let's consider a simple relayer contract as an example. As shown below, the relayer contract allows someone to make and sign a transaction, without having to execute the transaction. Often this is used when a user can't pay for the gas associated with the transaction.

The user who executes the transaction, the 'forwarder', can effectively censor transactions by using just enough gas so that the transaction executes, but not enough gas for the sub-call to succeed.