**ZafarKhan-K213567@nu.edu.pk**

## Introduction

This report details a critical vulnerability in MetaMask when interacting with ERC-20 tokens compiled with Solidity versions below 0.5.0. The vulnerability exploits leniency in input calldata length validation in these older Solidity versions, allowing malicious actors to craft deceptive transactions. The risk posed is significant, as it can lead to unauthorized token transfers or approvals.

## Description

**Background on Solidity and Calldata Handling:**
Solidity is the primary language for writing smart contracts on Ethereum. Each function call requires specific calldata structure, including the function signature (sighash) and parameters. For example, an ERC-20 transfer function expects:
**Function Signature:** 0xa9059cbb
**Parameters:**
-address to: 20 bytes
-uint256 value: 32 bytes

In Solidity versions prior to 0.5.0, the compiler does not enforce strict checks on calldata length. If the calldata is shorter than required, missing parts are filled with zero bytes (0x00).

**Implications:**
This can be exploited to craft transactions with truncated or malformed calldata that contracts still interpret as valid. MetaMask fails to handle these unusual transaction formats properly, leading to misleading transaction prompts.

**Exploitation Example:**
**1) Normal Transfer Call:**
**-Calldata:**
0xa9059cbb000000000000000000000000C588e338FdBB2CC523a1177f3D18e87FF5A16a6b0000000000000000000000000000000000000000000000000000000000989700
-**Interpreted As:**
address to: 0xC588e338FdBB2CC523a1177f3D18e87FF5A16a6b
uint256 value: 10000128

**2) Malformed Transfer Call:**
-Calldata:
0xa9059cbb000000000000000000000000C588e338FdBB2CC523a1177f3D18e87FF5A16a6b00000000000000000000000000000000000000000000000000000000009897
-**Interpreted As:**

address to: 0xC588e338FdBB2CC523a1177f3D18e87FF5A16a6b
uint256 value: 10000128 (truncated and padded with 0x00)

**ZafarKhan-K213567@nu.edu.pk**

Despite the missing byte, a contract compiled with Solidity < 0.5.0 will pad the calldata with zero bytes, causing the transaction to be processed normally without suspicion.

## Proof of Concept:

**Deploy a Vulnerable Contract:**
-Compile and deploy an ERC-20 token contract using Solidity 0.4.26.

**Initiate a Malformed Transaction:**
-Use a malicious script to create a transfer transaction with truncated calldata.

**MetaMask Interaction:**
-Connect to MetaMask on the Rinkeby network.
-Observe MetaMask prompt showing incomplete transaction details.

**Impact**

**User Deception:**
**-Phishing Attack Potential:** Attackers can create phishing websites that initiate malformed transactions. Users might see transaction prompts lacking critical details like the transfer amount or recipient address.
**-Incomplete Transaction Prompts:** MetaMask's inability to parse malformed calldata properly can mislead users into authorizing transactions without fully understanding their implications.

**Loss of Funds:**
**-Unauthorized Transfers:** Users may unknowingly approve transactions transferring tokens to attackers' addresses.
**-Unintended Approvals:** Attackers can gain approval for token transfers, allowing them to withdraw tokens from the user's wallet later