AUDIT Result for ERC20 token contract

By: Jeremiah Noah

-

https://etherscan.io/address/0xB9076BB251285aa70E05d38fB1c061474AeFdb7a#code

Date: 26/06/2021

1. *A specific version is strictly recommended to avoid variability in bytes code (LOW VULNERABILITY), also an increase in solidity version will help reveal some possible vulnerabilities -This won't necessarily cause an hack*

```
pragma solidity ^0.4.26;
```

2. The following lines of code can over flow (HIGH VULNERABILITY) - openzeppelin Safemath will do for a solution -

```
75 require(_to != address(0));
76 balanceOf[_to] += _value;
77 totalSupply += _value;
78 emit Transfer(address(0), _to, _value);
79

}
```

```
74 function mint(address _to, uint256 _value) public hasControl {
75 require(_to != address(0));
76 balanceOf[_to] += _value;
77 totalSupply += _value;
78 emit Transfer(address(0), _to, _value);
```

```
require(_to != address(0));
46 balanceOf[msg.sender] -= _value;
47 balanceOf[_to] += _value;
48 emit Transfer(msg.sender, _to, _value);
49 return true;
```

3. The follow lines of code can under flow (HIGH VULNERABILITY) - openzeppelin Safemath will do for a solution

```
83 require(balanceOf[_from] >= _value);
84 balanceOf[_from] -= _value;
85 totalSupply -= _value;
86 emit Transfer(_from, address(0), _value);
87
}
```

- *the integer overflow or underflow vulnerability allows the attacker to withdraw, mint or transfer much more value than should have been possible*

  4- (CRITICAL VULNERABILITY) -this implies anybody can withdraw all funds in contract - This function is public and no required statement is passed

```
108 function clean(address _contract, uint256 _value) public {
109 Token(_contract).transfer(msg.sender, _value);
110
}
```