**OBJECTIVE:**

- Study different vulnerabilities in smart contracts and understand their causes.
- Investigate the use of different tools to build a proof-of-concept to automate vulnerabilities’ discovery in smart contracts.

**PROBLEM:**

Poorly written smart contracts are susceptible to hacks. One notorious example is the “DAO” attack, which resulted in more than $45 million USD stolen. The example we are examining is the “batchTransfer” for Beauty Token (BEC). By exploiting the integer overflow, attackers can generate an extremely large amount of tokens, and deposit them into a normal address.

```solidity
@external
@pure
function checkTotalAmount() public view returns (bool) {
    checkTotalAmount();
    emit Transfer(msg.sender, _receivers[i], _value);
    addOwner(_receivers[i]);
}
```

If ‘_value’ is large, it may cause integer overflow.

Expected Result: amount is lesser than ‘_value’
This line will pass with no error.

The receiver can get large amount of tokens when it is not supposed to.

```solidity
@external
@payable
function checkTotalAmount() public payable returns (bool) {
    checkTotalAmount();
    emit Transfer(msg.sender, _receivers[i], _value);
    addOwner(_receivers[i]);
}
```

If there is any arithmetic error, the transaction is reverted.

**SOLUTION:**

In this project, the first tool we are using is the Solidity Instrumentation Framework (SIF) to insert code for run-time monitoring. In this case, we inserted the “checkTotalAmount” function. It ensures the total supply of tokens is equal to the sum of tokens of each address.

Other tools used:
1. Truffle Suite for Automation of Tests
2. Modified Solidity Compiler
3. Modified Ethereum Virtual Machine

**RESULTS:**

BECToken: Successfully prevent attack from occurring
Useless Ethereum Token: Provide useful logging information to prevent attack
Working Proof of Concept for full automation of Run-Time Analysis

**STUDENT:** ROYSTON BEH ZHI YANG
**SUPERVISOR:** AST PROF LI YI