18 types of bugs injection methods

Pnegcheng Zhang, Feng Xiao, Xiapu Luo, Hai Dong

Jan 2021

1 introduction

In our paper, we introduce the methods of injecting *re-entrancy* bugs and *integer overflow and underflow* bugs into *HuangGai* in detail. And we write this doc to help users understand how *HuangGai* injects 18 other types of bugs.

2 Predefined extraction criteria (*ContractExtractor*) and injection methods (*BugInjector*) of 18 types of bugs

1. Transaction order dependence.
   - Predefined extraction criteria (*ContractExtractor*): The contract needs to be developed based on *ERC-20 token standard* and contains the *approve* function.
   - injection methods (*BugInjector*): *HuangGai* invalidates security measures to allow the quota of the approved address is set from one nonzero value to another nonzero value, and label the assignment statement as a bug.
2. Results of contract execution affected by miners.
   - Predefined extraction criteria (*ContractExtractor*): *HuangGai* searches for the *if-statements* in a contract that meet the following conditions: When the type of the condition part of the *if-statement* is one of *bytes32*, *address payable*, *uint256*, or *address*, *HuangGai* will record the location and type of the *if-statement*. When such statements exist in a contract, *HuangGai* will be able to inject *results of contract execution affected by miners* bugs into the contract.
   - injection methods (*BugInjector*): *HuangGai* replaces an operand in the conditional part of the *if-statement* with the following global variables: *block.coinbase* (for address type), *block.coinbase* (for address payable type), *block.gaslimit* (for uint256 type), *block.number* (for

uint256 type), *block.timestamp* (for uint256 type), *blockhash(block.number)* (for bytes32 type), and label the *if-statement* as a bug.

3. Unhandled exception.

   - Predefined extraction criteria (*ContractExtractor*): The contract shall contain at least one of the following three types of statements: *call-statement*, *send-statement*, *delegatecall-statement*.
   - injection methods (*BugInjector*): *HuangGai* uses the following two ways to invalidate the security measures of low-level call statement (*call-statement*, *send-statement*, *delegatecall-statement*): receiving the return value but not checking the return value, or not receiving the return value. And *HuangGai* labels the *if-statement* as a bug.

4. Use *tx.origin* for authentication.

   - Predefined extraction criteria (*ContractExtractor*): *HuangGai* first captures the address type variables assigned in the *constructor* (we call these variables *ownerCandidate*) and then searches for bool expressions such as *ownerCandidate = = address type variable* or *ownercandidate != address type variable* in the contract. A contract needs to contain the bool expressions that meet the above conditions.
   - injection methods (*BugInjector*): *HuanggGai* replaces address type variable (no *ownerCandidate*) in the bool expression that meets the condition with *tx.origin* and labels the bool expression as a bug.

5. Wasteful contracts.

   - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain functions with the following conditions: the visibility is *public* or *external*, and the function needs to contain the statements that can transfer out ethers.
   - injection methods (*BugInjector*): *HuangGai* will invalidate all statements in the above functions (including the function modifiers used) that may interrupt the execution of these functions, and insert the statement to transfer out all ethers of the contract at the end of the function (*msg.sender.transfer (address (this.balance);*), and finally label the *transfer-statement* as a bug.

6. Short address attack.

   - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain functions that meet the following conditions: the visibility is *external* or *public*, and the function needs to contain statements that can transfer out ethers, the payee address and the number of ethers transferred are given by the function caller.

- injection methods (*BugInjector*): *HuangGai* will invalidate all *check statements* (i.e., statements that check *msg.data.length*) in the above functions (and the function modifiers used), and label these functions as bugs.

7. Suicide contracts.
   - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain functions that meet the following conditions: visibility is *external* or *public*, and functions need to contain *self-destruct-statements*.
   - injection methods (*BugInjector*): *HuangGai* will invalidate all statements in the above functions (and the function modifiers used) that may interrupt execution or authenticate, and label the *self-destruct-statements* (lines) as bugs.
8. Locked ether.
   - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain functions declared as *payable*.
   - injection methods (*BugInjector*): *HuangGai* will invalidate all the statements used to transfer out ethers in the contract in the following two ways: set the number of transferred out to 0, or change the *transfer-out-statements* to *comments*, and label the contract as bug.
9. Forced to receive ether.
   - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain *if-statements* (or *require-statements* or *assert-statements*) that meet the following conditions: the type of the conditional expression part of the statement is *uint256*, and at least one operand in the conditional expression part of the statement is *uint256 constant*.
   - injection methods (*BugInjector*): *HuangGai* will replace the *non-uint256* constant operand in the conditional expression part of the above statements with *address(this).balance*, and label these statements as bugs.
10. Pre-sent ether.
    - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain *if-statements* (or *require-statements* or *assert-statements*) that meet the following conditions: the type of the conditional expression part of the statement is *uint256*, and at least one operand in the conditional expression part of the statement is *uint256 constant*.
    - injection methods (*BugInjector*): *HuangGai* will replace the *non-uint256* constant operand in the conditional expression part of the above statements with *address(this).balance*, and label these statements as bugs.

11. Uninitialized local/state variables.

    - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain initialization statements for local variables or state variables (no constant).
    - injection methods (*BugInjector*): *HuangGai* will invalidate the assignments in these initialization statements and label these initialization statements as bugs.

12. Hash collisions with multiple variable length arguments.

    - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain functions that meet the following conditions: visibility is *external* or *public*, and functions contain multiple array type parameters, and function contains the declaration statement of the *bytes32* variable.
    - injection methods (*BugInjector*): *HuangGai* will re-assign the *bytes32* variables in the above functions to *keccak256(abi.encodePacked(parameters))*, where *parameters* are the array parameters passed in from outside, and label the *bytes32* assignment statement as bugs.

13. Specify *function* variable as any type.

    - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain *function* type variables.
    - injection methods (*BugInjector*): *HuangGai* will insert (assembly) statements so that external attackers can modify the type of function variables at will, and then label the inserted (assembly) statements as bugs.

14. Dos by complex *fallback* function.

    - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain the *fallback* functions that meet the following condition: the *fallback* function contains statements that can transfer out ethers.
    - injection methods (*BugInjector*): *HuangGai* will insert the statements at the end of the above *fallback* functions: *payee_address.call.gas(2301).value(1)("");* (in Solidity 0.5.x, 0.6.x) or *payee_ address.call{gas:2301, value:1}("");* (in Solidity 0.7.x), where the *payee_address* is the payee address in *fallback* function. After inserting the above statements, *HuangGai* labels the *fallback* functions as bugs.

15. *Public* function that could be declared *external*.

    - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain functions with *external* visibility.

- injection methods (*BugInjector*): *HuangGai* will change the visibility of the above functions from *external* to *public* and label these functions as bugs.

16. Non-public variables are accessed by *public*.

    - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain functions whose visibility is *external* or *public*.
    - injection methods (*BugInjector*): *HuangGai* will search for whether the above functions contain access operations to state variables with visibility of *private* or *internal*. If so, *HuangGai* will label these access operations as bugs.

17. Nonstandard naming.

    - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain at least one type of the following four types of structures: function, function modifier, event and constant variable.
    - injection methods (*BugInjector*): First, according to the naming standard of each structure, *HuangGai* changes the naming of these structures to non-standard form, then modifies the naming (including definition and use) of these structures in the contract by means of data flow tracing, and finally labels the declaration statements of structures as bugs.

18. Unlimited compiler versions.

    - Predefined extraction criteria (*ContractExtractor*): The contract needs to contain the version specification statement (pragma-solidity-statement).
    - injection methods (*BugInjector*): *HuangGai* will analyze these *pragma-solidity-statements* to get the minimum Solidity version that can compile the contract, and then replace the original *pragma-solidity-statements* with the following statement: *pragma solidity* $\wedge$*minimum_Solidity_version*, then label the new *pragma-solidity-statements* as bugs.