|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Basic rules – to ensure execution flow integrity** | | | | |
| Block Start | | **Expression** | **Rewritten code** | **Comment** |
| \_; | Check(cur\_block\_no);  \_; | Inject Check(); before the user code. |
| Block End | | **Expression** | **Rewritten code** | **Comment** |
| \_; (except for break, return, call) | \_;  Record(cur\_block\_no); | Inject Record(); after the user code. |
| break or return or call; | Record(cur\_block\_no);  break or return or call; | Inject Record(); before the user code where the end of code is break or return or call(). |
| **Extended rules – to ensure xxx** | | | | |
| - | **Statement** | | **Rewritten code** | **Comment** |
| a = x + y; | | If (CheckIntAdd (x,y))  a = IntAdd(x,y);  else revert(); | This rule supports other arithmetic operations (+,/, %, and so on). |
| a--; | | If (CheckUnaryMinus(a))  a = CheckUnaryMinus(a);  else revert(); | This rule also supports prefix increment operation(++). |
| for (int x = 0; x < y; x++)  {  \_;  } | | for (int x = 0; x < y; x++)  {  if (checkGasLimit(y, cur\_block\_no))  {  \_;  }  else revert();  } | This rule checks that the approximate gas limit to execute this loop is safe or not. For this our tool pre-calculates the gas fee for each basic block. |
| a = x.call() + \_; | | tmp = x.call();  << divide call execution  a = tmp + \_; | This rule separates the assignment statement to external call() and some operations. |