

```

1   $\exists \text{ arg}_0, \text{ arg}_1, r_1, r_2, r_3, \text{ call}$ 
2  ( $\&\& \text{ (= } r_3 \text{ (}\otimes \text{ } r_1 \text{ } r_2\text{))}$ )
3    ( $> \llbracket r_2 \rrbracket \llbracket r_3 \rrbracket$ )
4    ( $\text{interfere? } r_2 \text{ call.value}$ )
5    ( $\text{interfere? } \text{arg}_0 \text{ call.addr}$ )
6    ( $\text{interfere? } \text{arg}_1 \text{ call.value}$ ))
7  where  $\otimes \in \{+, \times\}$ 

```

(a) Query for the BatchOverflow

```

1  batchTransfer( $\text{arg}_0$ ,  $\text{arg}_1$ ) {
2    ...
3     $r_3 = r_1 \otimes r_2$ ;
4    ...
5    call(gas, addr, value, ...);
6    ...
7  }

```

The diagram illustrates the key pattern of the BatchOverflow vulnerability. It shows a `batchTransfer` function call with arguments `arg0` and `arg1`. Inside the function, `r3` is assigned the result of `r1` \otimes `r2`. The `call` function is then invoked with `gas`, `addr`, `value`, and other arguments. Red arrows indicate the flow of data: `arg0` points to `addr` in the `call` function, `arg1` points to `value` in the `call` function, and the `overflow` symbol points to `value` in the `call` function. The word `interfere` is also present near the `value` parameter.

(b) The key pattern of the BatchOverflow Vulnerability