CollAFL : Path Sensitive Fuzzing

S & P 2018

Coverage

For man is man and
master of his fate.

Block Coverage

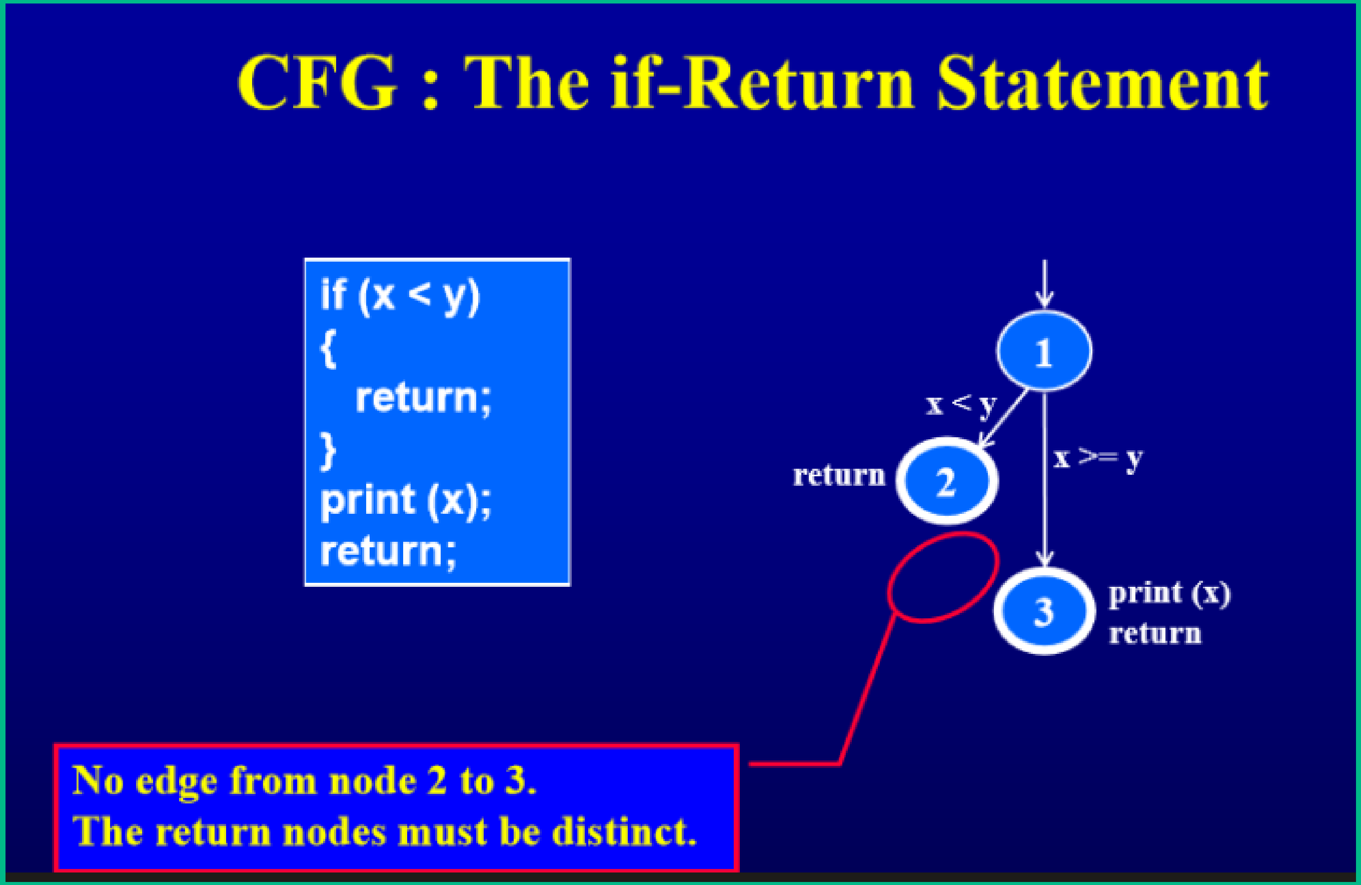
A sequence of statements such that if the first statement is executed, all statements will be (no branches).

Edge Coverage

Transfers of control.

Path Coverage

A sequence of edge.



CFG : The if-Return Statement

```
if (x < y)
{
   return;
}
print (x);
return;
```

No edge from node 2 to 3.
The return nodes must be distinct.

AFL

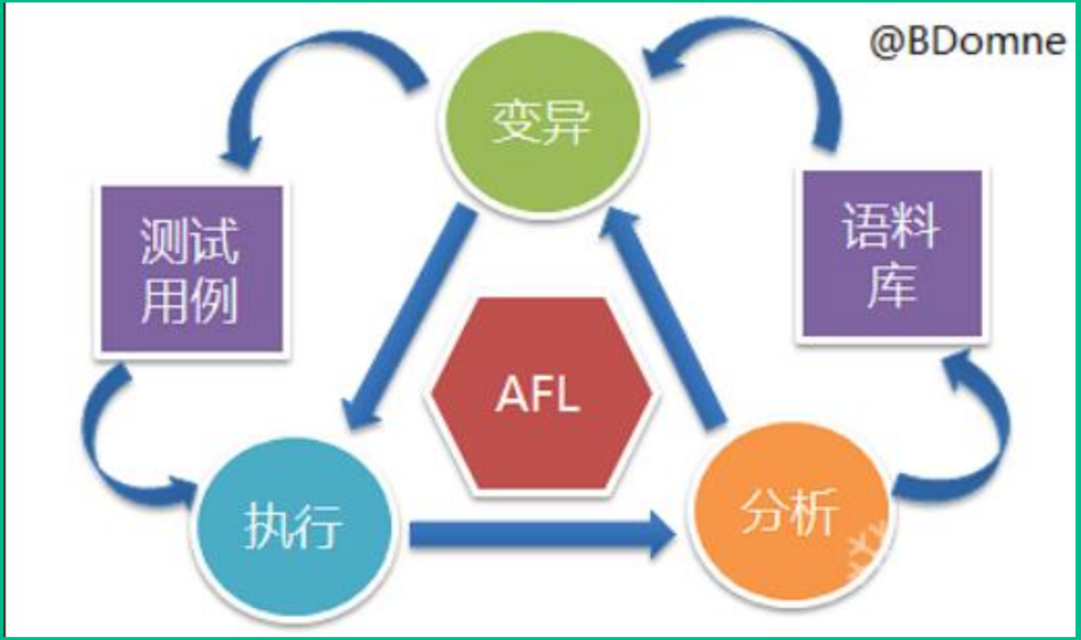
For man is man and
master of his fate.

Coverage-guided fuzzing

Tracking code coverage?

AFL(in GCC and LLVM mode) uses static instrumentation with a compact bitmap（64KB） to track edge coverage

当有新的 edge 出现或已有 edge 中出现新的命中组则视为产生新状态，相应的测试用例将被归入到语料库中



AFL

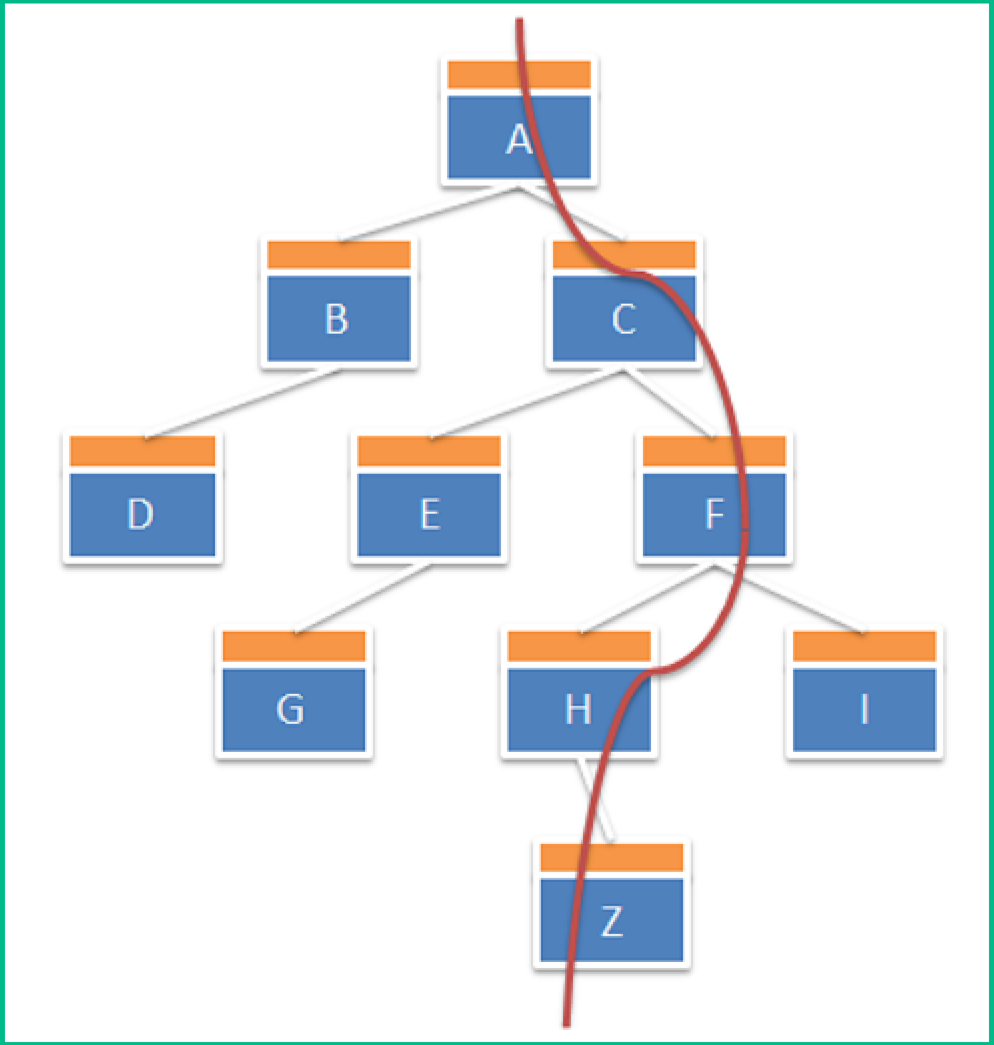
For man is man and
master of his fate.

蓝色块代表程序执行过程中的基本块

黄色块代表相应的用于统计的探针代码

4 个 tuple（AC，CF，FH，HZ）

命中次数各一次



AFL

For man is man and
master of his fate.

快速判断当前的测试用例有没有使得系统产生新的状态

How to save edge coverage in 64KB Bitmap?

Hash: $cur \oplus (prev \gg 1)$

Hash Value	Hit Count

edge : A -> B

cur : the key of B

prev : the key of A

Hash Collision

For man is man and
master of his fate.

Can we increase the size of bitmap?

64KB -> 4MB : 60% drop-off

CollAFL’s Solution to Hash Collision

For man is man and
master of his fate.

为每个block分配一个x, z

一条路径上的block共用一个y

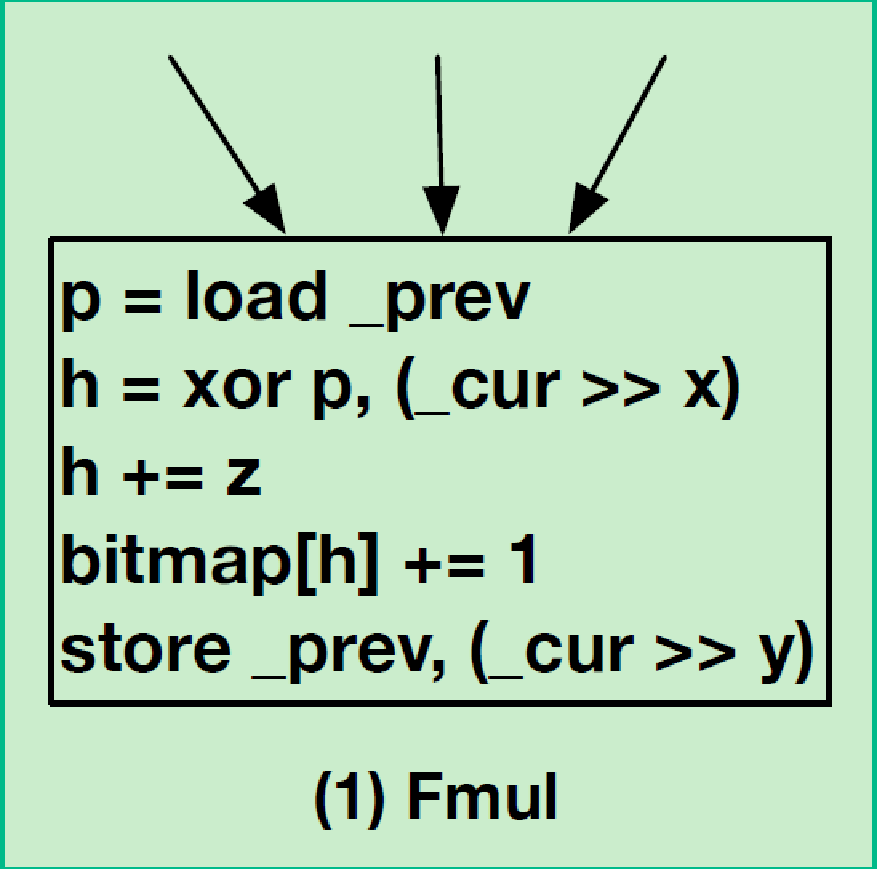
Hash: $Fmul(cur, prev) = (cur \gg x) \oplus (prev \gg y) + z$

edge : A -> B

cur : the key of B

prev : the key of A

parameter : <x, y, z>


```
p = load _prev
h = xor p, (_cur >> x)
h += z
bitmap[h] += 1
store _prev, (_cur >> y)
```

(1) Fmul

CollAFL's Solution to Hash Collision

For man is man and
master of his fate.

Try to find a solution of <x, y, z> for each block → **Once found** → **Resolve the hash collision**

But no guarantee

Single Precedent

For man is man and
master of his fate.

超过60%的block只有一个
precedent

提高Fuzzer的吞吐量

Hash: $Fsingle(cur, prev) : c$

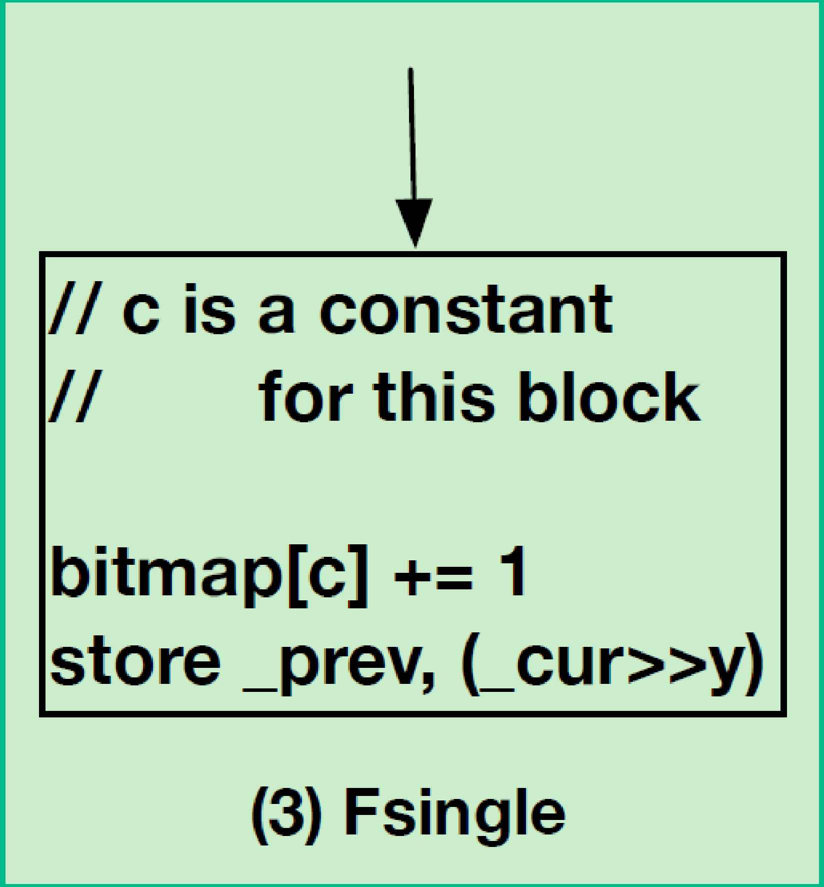
edge : A -> B

cur : the key of B

prev : the key of A

c : unique constant

hard-coded in B



```
// c is a constant
//        for this block

bitmap[c] += 1
store _prev, (_cur>>y)
```

(3) Fsingle

Multiple Precedents

For man is man and
master of his fate.

Unsolvable Block

Hash: $Fhash(cur, prev) : hash\_table\_lookup(cur, prev)$

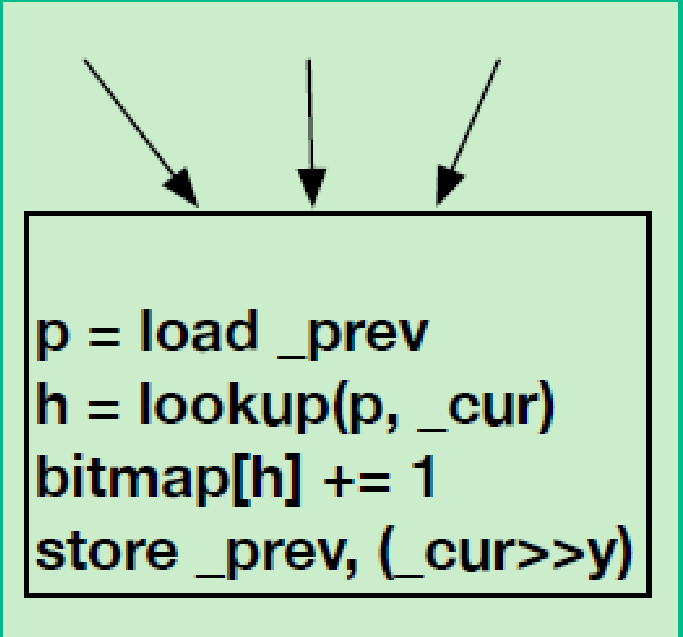
edge : A -> B

cur : the key of B

prev : the key of A

Hash table offline

Precomputed!



```
p = load _prev
h = lookup(p, _cur)
bitmap[h] += 1
store _prev, (_cur>>y)
```

(2) Fhash

Solution

For man is man and
master of his fate.

$$F = \begin{cases} Fmul, & \texttt{Solvable blocks with multi pred} \\ Fhash, & \texttt{Unsolvable blocks with ...} \\ Fsingle, & \texttt{Blocks with single precedent} \end{cases}$$

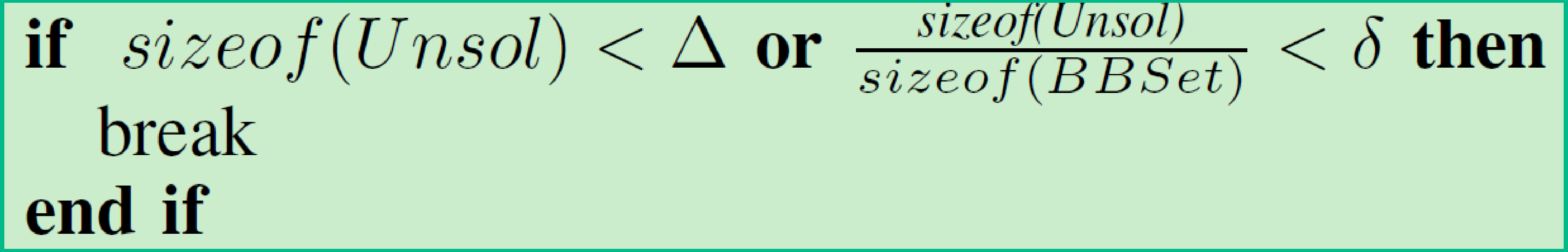
Static analysis tool or compiler → **Retrieve blocks and precedent information** → **Assign unique random keys to each block**

Solution

For man is man and
master of his fate.

遍历x, y, z的可能取值，
找到一个比较好的解就可以
停了

Greedy algorithm to get <x, y, z>



```
if $sizeof(Unsol) < \Delta$ or $\frac{sizeof(Unsol)}{sizeof(BBSet)} < \delta$ then
    break
end if
```

Solution

For man is man and
master of his fate.

对于unsolved block, 从bitmap中随机选没用过的hash值即可

Unsolved block

$$FreeHashes = BITMAP\_HASHES - Hashes$$

$$FreeHashes.\mathbf{RandomPop}()$$

Prioritize seed selection

For man is man and
master of his fate.

Untouched-neighbor-branch guided policy

$$Weight\_Br(T) = \sum_{\substack{bb \in Path(T) \\ <bb, bb_i> \in EDGES}} IsUntouched(<bb, bb_i>)$$

Prioritize seed selection

For man is man and
master of his fate.

Untouched-neighbor-descendant guided policy

$$Weight\_Desc(T) = \sum_{\substack{bb \in Path(T) \\ IsUntouched(<bb, bb_i>)}} NumDesc(bb_i)$$

Prioritize seed selection

For man is man and
master of his fate.

Memory-access guided policy

$$Weight\_Mem(T) = \sum_{bb \in Path(T)} NumMemInstr(bb)$$

THANKS !