# BUG b-36

Location：readelf.c:5211 process\_dynamic\_segment() if (\*name)

Analysis:

The code fragment at the crash location is as follows:

static int process\_dynamic\_segment (file)

{

if (dynamic\_size == 0)

return 1;

if (is\_32bit\_elf)

return 0;

if (dynamic\_symbols == NULL)

……

if (dynamic\_strings == NULL)

……

if (dynamic\_syminfo == NULL)

……

for (i = 0, entry = dynamic\_segment; i < dynamic\_size; i++, entry++)

switch (entry->d\_tag)

{

……

case DT\_FINI\_ARRAY:

if (do\_dynamic)

{

if (dynamic\_strings != NULL && entry->d\_tag == DT\_USED)

{

char \*name;

name = dynamic\_strings + entry->d\_un.d\_val;

if (\*name) //crash

It is clear that this bug is behind a couple of nested condition statements and therefore triggering it needs to satisfy 7 conditions, each of which can be affected by different parts of the seed input.

FairFuzz proposes to record the mutation operations/locations that do not affect “rare” branches, most of which leads to the deeper part of the PUT. It is obvious that this technique can helps pass the nested condition statements, because it keeps the bytes in the seed inputs that pass the shallower condition statements unchanged.

On the other hand, FunAFL focuses more efforts on the seed selection and inherits AFL’s mutation strategy. So even FunAFL finds a seed input that have the correct bytes satisfying the first few condition statement and keeps it in the seed queue, there is still probabilities that these bytes are mutated to other values and the mutated seed input fails to pass the condition statements that the original seed input can pass.

Among the 5 repetitions, AFL triggers this crash only once and costs 18.87 hours to find it. We owe this one-time discovery to the randomness of the havoc mode in AFL’s mutation strategy.

It is worth noting that FairFuzz’s improvement on mutation strategy is orthogonal to FunAFL’s seed selection strategy. It can be expected that integrating the two techniques achieves even better performance in bug detection. And it is part of our future works.