author	Youlin Li <liulin063@gmail.com>	2022-11-03 17:34:39 +0800
committer	Daniel Borkmann <daniel@iogearbox.net>	2022-11-04 00:24:12 +0100
commit	f1db20814af532f85e091231223e5e4818e8464b (patch)	
tree	fd056852b9da624c5355d39a81032e9410053c36	
parent	8bbabb3fddcd0f858be69ed5abc9b470a239d6f2 (diff)	
download	linux-f1db20814af532f85e091231223e5e4818e8464b.tar.gz	

bpf: Fix wrong reg type conversion in release_reference()

Some helper functions will allocate memory. To avoid memory leaks, the verifier requires the eBPF program to release these memories by calling the corresponding helper functions.

When a resource is released, all pointer registers corresponding to the resource should be invalidated. The verifier use release_references() to do this job, by apply __mark_reg_unknown() to each relevant register.

It will give these registers the type of SCALAR_VALUE. A register that will contain a pointer value at runtime, but of type SCALAR_VALUE, which may allow the unprivileged user to get a kernel pointer by storing this register into a map.

Using __mark_reg_not_init() while NOT allow_ptr_leaks can mitigate this problem.

Fixes: fd978bf7fd31 ("bpf: Add reference tracking to verifier")
Signed-off-by: Youlin Li <liulin063@gmail.com>
Signed-off-by: Daniel Borkmann <daniel@iogearbox.net>
Link: https://lore.kernel.org/bpf/20221103093440.3161-1-liulin063@gmail.com

Diffstat

-rw-r--r-- kernel/bpf/verifier.c 8

1 files changed, 6 insertions, 2 deletions

```
diff --git a/kernel/bpf/verifier.c b/kernel/bpf/verifier.c
index dd9019c8b0db04..225666307bba39 100644
--- a/kernel/bpf/verifier.c
+++ b/kernel/bpf/verifier.c
@@ -6623,8 +6623,12 @@ static int release_reference(struct bpf_verifier_env *env,
                return err;

        bpf_for_each_reg_in_vstate(env->cur_state, state, reg, ({
-               if (reg->ref_obj_id == ref_obj_id)
-                       __mark_reg_unknown(env, reg);
+               if (reg->ref_obj_id == ref_obj_id) {
+                       if (!env->allow_ptr_leaks)
+                               __mark_reg_not_init(env, reg);
+                       else
+                               __mark_reg_unknown(env, reg);
+               }
```

```
}));

return 0;
```