BPF Selftest Notes

General instructions on running selftests can be found in Documentation/bpf/bpf_devel_QA.rst.

Running Selftests in a VM

It's now possible to run the selftests using `tools/testing/selftests/bpf/vmtest.sh`. The script tries to ensure that the tests are run with the same environment as they would be run post-submit in the CI used by the Maintainers.

This script downloads a suitable Kconfig and VM userspace image from the system used by the CI. It builds the kernel (without overwriting your existing Kconfig), recompiles the bpf selftests, runs them (by default `tools/testing/selftests/bpf/test_progs`) and saves the resulting output (by default in `~/.bpf_selftests`).

Script dependencies: - clang (preferably built from sources, https://github.com/llvm/llvm-project); - pahole (preferably built from sources, https://git.kernel.org/pub/scm/devel/pahole/pahole.git/); - qemu; - docutils (for `rst2man`); - libcap-devel.

For more information on about using the script, run:

```
$ tools/testing/selftests/bpf/vmtest.sh -h
```

In case of linker errors when running selftests, try using static linking:

```
$ LDLIBS=-static vmtest.sh
```

> **Note**
>
> Some distros may not support static linking.

> **Note**
>
> The script uses pahole and clang based on host environment setting. If you want to change pahole and llvm, you can change *PATH* environment variable in the beginning of script.

> **Note**
>
> The script currently only supports x86_64 and s390x architectures.

Additional information about selftest failures are documented here.

profiler[23] test failures with clang/llvm <12.0.0

With clang/llvm <12.0.0, the profiler[23] test may fail. The symptom looks like

```
// r9 is a pointer to map_value
// r7 is a scalar
17:       bf 96 00 00 00 00 00 00 r6 = r9
18:       0f 76 00 00 00 00 00 00 r6 += r7
math between map_value pointer and register with unbounded min value is not allowed

// the instructions below will not be seen in the verifier log
19:       a5 07 01 00 01 01 00 00 if r7 < 257 goto +1
20:       bf 96 00 00 00 00 00 00 r6 = r9
// r6 is used here
```

The verifier will reject such code with above error. At insn 18 the r7 is indeed unbounded. The later insn 19 checks the bounds and the insn 20 undoes map_value addition. It is currently impossible for the verifier to understand such speculative pointer arithmetic. Hence this patch addresses it on the compiler side. It was committed on llvm 12.

The corresponding C code

```
for (int i = 0; i < MAX_CGROUPS_PATH_DEPTH; i++) {
        filepart_length = bpf_probe_read_str(payload, ...);
        if (filepart_length <= MAX_PATH) {
                barrier_var(filepart_length); // workaround
                payload += filepart_length;
        }
}
```

bpf_iter test failures with clang/llvm 10.0.0

With clang/llvm 10.0.0, the following two bpf_iter tests failed:

- `bpf_iter/ipv6_route`

- `bpf_iter/netlink`

The symptom for `bpf_iter/ipv6_route` looks like

```
2: (79) r8 = *(u64 *)(r1 +8)
...
14: (bf) r2 = r8
15: (0f) r2 += r1
; BPF_SEQ_PRINTF(seq, "%pi6 %02x ", &rt->fib6_dst.addr, rt->fib6_dst.plen);
16: (7b) *(u64 *)(r8 +64) = r2
only read is supported
```

The symptom for `bpf_iter/netlink` looks like

```
; struct netlink_sock *nlk = ctx->sk;
2: (79) r7 = *(u64 *)(r1 +8)
...
15: (bf) r2 = r7
16: (0f) r2 += r1
; BPF_SEQ_PRINTF(seq, "%pK %-3d ", s, s->sk_protocol);
17: (7b) *(u64 *)(r7 +0) = r2
only read is supported
```

This is due to a llvm BPF backend bug. The fix has been pushed to llvm 10.x release branch and will be available in 10.0.1. The patch is available in llvm 11.0.0 trunk.

bpf_verif_scale/loop6.o test failure with Clang 12

With Clang 12, the following bpf_verif_scale test failed:

- `bpf_verif_scale/loop6.o`

The verifier output looks like

```
R1 type=ctx expected=fp
The sequence of 8193 jumps is too complex.
```

The reason is compiler generating the following code

```
;       for (i = 0; (i < VIRTIO_MAX_SGS) && (i < num); i++) {
    14:       16 05 40 00 00 00 00 00 if w5 == 0 goto +64 <LBB0_6>
    15:       bc 51 00 00 00 00 00 00 w1 = w5
    16:       04 01 00 00 ff ff ff ff w1 += -1
    17:       67 05 00 00 20 00 00 00 r5 <<= 32
    18:       77 05 00 00 20 00 00 00 r5 >>= 32
    19:       a6 01 01 00 05 00 00 00 if w1 < 5 goto +1 <LBB0_4>
    20:       b7 05 00 00 06 00 00 00 r5 = 6
00000000000000a8 <LBB0_4>:
    21:       b7 02 00 00 00 00 00 00 r2 = 0
    22:       b7 01 00 00 00 00 00 00 r1 = 0
;       for (i = 0; (i < VIRTIO_MAX_SGS) && (i < num); i++) {
    23:       7b 1a e0 ff 00 00 00 00 *(u64 *)(r10 - 32) = r1
    24:       7b 5a c0 ff 00 00 00 00 *(u64 *)(r10 - 64) = r5
```

Note that insn #15 has w1 = w5 and w1 is refined later but r5(w5) is eventually saved on stack at insn #24 for later use. This cause later verifier failure. The bug has been fixed in Clang 13.

BPF CO-RE-based tests and Clang version

A set of selftests use BPF target-specific built-ins, which might require bleeding-edge Clang versions (Clang 12 nightly at this time).

Few sub-tests of core_reloc test suit (part of test_progs test runner) require the following built-ins, listed with corresponding Clang diffs introducing them to Clang/LLVM. These sub-tests are going to be skipped if Clang is too old to support them, they shouldn't cause build failures or runtime test failures:

- __builtin_btf_type_id() [0_, 1_, 2];
- __builtin_preserve_type_info(), __builtin_preserve_enum_value() [3, 4].

Floating-point tests and Clang version

Certain selftests, e.g. core_reloc, require support for the floating-point types, which was introduced in Clang 13. The older Clang versions will either crash when compiling these tests, or generate an incorrect BTF.

Kernel function call test and Clang version

Some selftests (e.g. kfunc_call and bpf_tcp_ca) require a LLVM support to generate extern function in BTF. It was introduced in Clang 13.

Without it, the error from compiling bpf selftests looks like:

```
libbpf: failed to find BTF for extern 'tcp_slow_start' [25] section: -2
```

btf_tag test and Clang version

The btf_tag selftest requires LLVM support to recognize the btf_decl_tag and btf_type_tag attributes. They are introduced in *Clang 14* [0_, 1_]. The subtests `btf_type_tag_user_{mod1, mod2, vmlinux}` also requires pahole version `1.23`.

Without them, the btf_tag selftest will be skipped and you will observe:

```
#<test_num> btf_tag:SKIP
```

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\tools\testing\selftests\bpf\(linux-master)(tools)(testing)(selftests)(bpf)README.rst, line 226); *backlink*

Duplicate explicit target name: "0".

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\tools\testing\selftests\bpf\(linux-master)(tools)(testing)(selftests)(bpf)README.rst, line 227); *backlink*

Duplicate explicit target name: "1".

Clang dependencies for static linking tests

linked_vars, linked_maps, and linked_funcs tests depend on Clang fix to generate valid BTF information for weak variables. Please make sure you use Clang that contains the fix.

Clang relocation changes

Clang 13 patch clang reloc patch made some changes on relocations such that existing relocation types are broken into more types and each new type corresponds to only one way to resolve relocation. See kernel llvm reloc for more explanation and some examples. Using clang 13 to compile old libbpf which has static linker support, there will be a compilation failure:

```
libbpf: ELF relo #0 in section #6 has unexpected type 2 in .../bpf_tcp_nogpl.o
```

Here, `type 2` refers to new relocation type `R_BPF_64_ABS64`. To fix this issue, user newer libbpf.

Clang dependencies for the u32 spill test (xdpwall)

The xdpwall selftest requires a change in Clang 14.

Without it, the xdpwall selftest will fail and the error message from running test_progs will look like:

```
test_xdpwall:FAIL:Does LLVM have https://reviews.llvm.org/D109073? unexpected error: -4007
```

Docutils System Messages

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\tools\testing\selftests\bpf\(linux-master)(tools)(testing)(selftests)(bpf)README.rst, line 180); *backlink*

Duplicate target name, cannot be used as a unique reference: "0".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\tools\testing\selftests\bpf\(linux-master)(tools)(testing)(selftests)(bpf)README.rst, line 180); *backlink*

Duplicate target name, cannot be used as a unique reference: "1".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\tools\testing\selftests\bpf\(linux-master)(tools)(testing)(selftests)(bpf)README.rst, line 215); *backlink*

Duplicate target name, cannot be used as a unique reference: "0".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\tools\testing\selftests\bpf\(linux-master)(tools)(testing)(selftests)

(bpf)README.rst, line 215); *backlink*

Duplicate target name, cannot be used as a unique reference: "1".