Date: Tue, 14 Jun 2022 13:05:08 +1000
From: Michael Ellerman <mpe@...erman.id.au>
To: oss-security@...ts.openwall.com
Subject: CVE-2022-32981: Linux kernel for powerpc 32-bit, buffer overflow in
 ptrace PEEKUSER/POKEUSER

The Linux kernel for powerpc 32-bit has a buffer overflow in the handling of ptrace
PEEKUSER/POKEUSER when accessing floating point registers.

The fix for mainline is:
  https://git.kernel.org/pub/scm/linux/kernel/git/powerpc/linux.git/commit/?
id=8e1278444446fc97778a5e5c99bca1ce0bbc5ec9

Which is included in v5.19-rc2.

Stable backports have been posted.

A test case is included below, it will report if the system is correctly
patched, it is safe to run on an unpatched system.

cheers

```
#undef NDEBUG
#include <assert.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/ptrace.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/syscall.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>

static double expected = 0.123456L;

static int child(int shm_id)
{
        int *cptr = shmat(shm_id, NULL, 0);

        asm volatile (
                "lfd     %%f0, 0(%0)    ;"
                "lfd     %%f1, 0(%0)    ;"
                "li      %%r9, 1        ;"
                "stw     %%r9, 0(%1)    ;"
                "1:"
                "lwz     %%r9, 0(%2)    ;"
                "cmpwi   %%r9, 0        ;"
                "beq       1b           ;"
                : // outputs
                : // inputs
                  "b" (&expected), "b" (&cptr[1]), "b" (&cptr[0])
                : // clobbers
                  "memory", "r9", "fr0", "fr1"
        );

        return 0;
}

int start_trace(pid_t child)
{
        int ret;
```

```
        ret = ptrace(PTRACE_ATTACH, child, NULL, NULL);
        if (ret) {
                perror("ptrace(PTRACE_ATTACH) failed");
                return -1;
        }
        ret = waitpid(child, NULL, 0);
        if (ret != child) {
                perror("waitpid() failed");
                return -1;
        }
        return 0;
}

int stop_trace(pid_t child)
{
        int ret;

        ret = ptrace(PTRACE_DETACH, child, NULL, NULL);
        if (ret) {
                perror("ptrace(PTRACE_DETACH) failed");
                return -1;
        }
        return 0;
}

long raw_ptrace(enum __ptrace_request request, pid_t pid, unsigned long addr, void *data)
{
        return syscall(__NR_ptrace, request, pid, (void *)addr, data);
}

#define PEEKS_PER_FPR   (sizeof(__u64) / sizeof(unsigned long))

int peek_fpr(pid_t child, int frnum, __u64 *fpr)
{
        unsigned long *p, addr;
        int i, fpindex;
        long ret;

        fpindex = PEEKS_PER_FPR * frnum;

        p = (unsigned long *)fpr;
        for (i = 0; i < PEEKS_PER_FPR; i++, p++) {
                addr = sizeof(unsigned long) * (PT_FPR0 + fpindex + i);
                ret = raw_ptrace(PTRACE_PEEKUSER, child, addr, p);
                if (ret) {
                        perror("ptrace(PTRACE_PEEKUSR) failed");
                        return -1;
                }
        }

        return 0;
}

int parent(pid_t child)
{
        double f0, f1;

        assert(start_trace(child) == 0);

        assert(peek_fpr(child, 0, (__u64 *)&f0) == 0);
        assert(peek_fpr(child, 1, (__u64 *)&f1) == 0);

        assert(stop_trace(child) == 0);

        printf("expected = %e\n", f0);
        printf("f0       = %e\n", f0);
        printf("f1       = %e\n", f1);

        if (f0 != expected || f1 != expected) {
                printf("FAIL - values don't match! Kernel is buggy.\n");
                return -1;
        }

        printf("OK - values match\n");
```

```
        return 0;
}

int main(void)
{
        int shm_id, ret, status, *pptr;
        pid_t pid;

        shm_id = shmget(IPC_PRIVATE, sizeof(int) * 2, 0777|IPC_CREAT);
        assert(shm_id != -1);

        pid = fork();
        assert(pid >= 0);

        if (pid == 0)
                exit(child(shm_id));

        pptr = shmat(shm_id, NULL, 0);

        // Wait for child to signal us to continue
        while (!pptr[1])
                asm volatile("" : : : "memory");

        ret = parent(pid);
        if (ret) {
                kill(pid, SIGTERM);
                shmdt((void *)pptr);
                shmctl(shm_id, IPC_RMID, NULL);
                return -1;
        }

        // Signal child to exit
        pptr[0] = 1;
        shmdt((void *)pptr);

        ret = wait(&status);
        shmctl(shm_id, IPC_RMID, NULL);

        assert(ret != -1 && WIFEXITED(status) && !WEXITSTATUS(status));

        return 0;
}
```