Stack overflow via recursive loop in Gawk

Description

GAWK is a widely used domain-specific language designed for text processing. The "parse_reg_exp" function in "support/regcomp.c" may trigger stack overflow via recursive loop. The vulnerability involves function "parse_expression", "parse_branch" and "parse_sub_exp" and exists in latest stable release (gawk-5.1.1) and latest master branch (12f0f4f27872cd4df6c63977fd0c6ff63d29e424, updated on July 29, 2022).The vulnerable code (located at support/regcomp.c) and the bug's basic introduction are highlighted as follows:

```
// support/regcomp.c
static bin_tree_t *
parse_reg_exp (re_string_t *regexp, regex_t *preg, re_token_t *token,
           reg_syntax_t syntax, Idx nest, reg_errcode_t *err)
{
  re_dfa_t *dfa = preg->buffer;
  bin_tree_t *tree, *branch = NULL;
  bitset_word_t initial_bkref_map = dfa->completed_bkref_map;
    // line 2121
    // under this poc, the function has recursive loop which leads to stack
overflow
  tree = parse_branch (regexp, preg, token, syntax, nest, err);
  if (__glibc_unlikely (*err != REG_NOERROR && tree == NULL))
    return NULL;

  while (token->type == OP_ALT)
    {
      fetch_token (token, regexp, syntax | RE_CARET_ANCHORS_HERE);
      if (token->type != OP_ALT && token->type != END_OF_RE
      && (nest == 0 || token->type != OP_CLOSE_SUBEXP))
    {
      bitset_word_t accumulated_bkref_map = dfa->completed_bkref_map;
      dfa->completed_bkref_map = initial_bkref_map;
    // line 2133
    // under this poc, the function has recursive loop which leads to stack
overflow
      branch = parse_branch (regexp, preg, token, syntax, nest, err);
      if (__glibc_unlikely (*err != REG_NOERROR && branch == NULL))
        {
          if (tree != NULL)
        postorder (tree, free_tree, NULL);
          return NULL;
        }
```

Proof of Concept

Build the gawk (5.1.1 or latest commit 12f0f4f27872cd4df6c63977fd0c6ff63d29e424) and run it using the input POC.

```
# build the gawk with address sanitizer
export CFLAGS=" -fsanitize=address "; export CXXFLAGS=" -fsanitize=address ";
export LDFLAGS=" -fsanitize=address ";
CFGARG=" --enable-shared=no "; configure

AFL_USE_ASAN=1 LD=afl-gcc--disable-shared make

# the INPUT_FILE is not decesive, which can be any valid input.
./gawk -f POC_FILE INPUT_FILE
```

The stack dump is:

```
Disassembly:
Stack Head (1000 entries):
   parse_expression          @ 0x0000555555704fb9: in /src/AWKs/gawk-5.1.1/gawk
   parse_branch              @ 0x000055555570a1fc: in /src/AWKs/gawk-5.1.1/gawk
   parse_reg_exp             @ 0x000055555570a783: in /src/AWKs/gawk-5.1.1/gawk
   parse_sub_exp             @ 0x0000555555707093: in /src/AWKs/gawk-5.1.1/gawk
   parse_expression          @ 0x0000555555707093: in /src/AWKs/gawk-5.1.1/gawk
   parse_branch              @ 0x000055555570a1fc: in /src/AWKs/gawk-5.1.1/gawk
   parse_reg_exp             @ 0x000055555570a783: in /src/AWKs/gawk-5.1.1/gawk
   parse_sub_exp             @ 0x0000555555707093: in /src/AWKs/gawk-5.1.1/gawk
   parse_expression          @ 0x0000555555707093: in /src/AWKs/gawk-5.1.1/gawk
   parse_branch              @ 0x000055555570a1fc: in /src/AWKs/gawk-5.1.1/gawk
   parse_reg_exp             @ 0x000055555570a783: in /src/AWKs/gawk-5.1.1/gawk
   parse_sub_exp             @ 0x0000555555707093: in /src/AWKs/gawk-5.1.1/gawk
   parse_expression          @ 0x0000555555707093: in /src/AWKs/gawk-5.1.1/gawk
   parse_branch              @ 0x000055555570a1fc: in /src/AWKs/gawk-5.1.1/gawk
   parse_reg_exp             @ 0x000055555570a783: in /src/AWKs/gawk-5.1.1/gawk
   parse_sub_exp             @ 0x0000555555707093: in /src/AWKs/gawk-5.1.1/gawk
Registers:
rax=0x0000555555975700 rbx=0x00007ffff7770010 rcx=0x000000000023b24d
rdx=0x00007fffffffd6c0
rsi=0x0000555555975580 rdi=0x00007fffffffd6e0 rbp=0x00007fffffffd6c0
rsp=0x00007fffff7ff000
 r8=0x00000000000031e9  r9=0x00007fffffffd6bc r10=0x00007fffffffd6c0
r11=0x00007ffff7770010
r12=0x00007fffffffd6c0 r13=0x00000000000031e9 r14=0x00007ffff7770010
r15=0x0000555555975700
rip=0x0000555555704fb9 efl=0x0000000000010246  cs=0x0000000000000033
 ss=0x000000000000002b
 ds=0x0000000000000000  es=0x0000000000000000  fs=0x0000000000000000
 gs=0x0000000000000000
 k0=0x0000000000000000  k1=0x0000000000000000  k2=0x000000005b77ffff
 k3=0x0000000000000000
 k4=0x0000000000000000  k5=0x0000000000000000  k6=0x0000000000000000
 k7=0x0000000000000000

#0  0x00005555558cb81e in peek_token (token=0x0, input=0x0, syntax=0)
    at ./regcomp.c:1744
#1  0x00005555558ca919 in fetch_token (result=0x7fffffff8e60,
    input=0x7fffffff8ff0, syntax=10728013) at ./regcomp.c:1736
#2  0x00005555558d4807 in parse_sub_exp (regexp=0x7fffffff8ff0,
    preg=0x60b000000880, token=0x7fffffff8e60, syntax=2339405,
```

```
    nest=4429, err=0x7fffffff8fe0) at ./regcomp.c:2449
#3  0x00005555558d1c57 in parse_expression (regexp=0x7fffffff8ff0,
    preg=0x60b000000880, token=0x7fffffff8e60, syntax=2339405,
    nest=4428, err=0x7fffffff8fe0) at ./regcomp.c:2242
#4  0x00005555558d03e1 in parse_branch (regexp=0x7fffffff8ff0,
    preg=0x60b000000880, token=0x7fffffff8e60, syntax=2339405,
    nest=4428, err=0x7fffffff8fe0) at ./regcomp.c:2169
#5  0x00005555558caa4c in parse_reg_exp (regexp=0x7fffffff8ff0,
    preg=0x60b000000880, token=0x7fffffff8e60, syntax=2339405,
    nest=4428, err=0x7fffffff8fe0) at ./regcomp.c:2121
......
#17716 0x00005555558d076f in parse_branch (regexp=0x7fffffff8ff0,
preg=0x60b000000880, token=0x7fffffff8e60, syntax=2339405, nest=0,
err=0x7fffffff8fe0) at ./regcomp.c:2176
#17717 0x00005555558caeec in parse_reg_exp (regexp=0x7fffffff8ff0,
preg=0x60b000000880, token=0x7fffffff8e60, syntax=2339405, nest=0,
err=0x7fffffff8fe0) at ./regcomp.c:2133
#17718 0x00005555558c02a9 in parse (regexp=0x7fffffff8ff0, preg=0x60b000000880,
syntax=2339405, err=0x7fffffff8fe0) at ./regcomp.c:2089
#17719 0x00005555558b1828 in re_compile_internal (preg=0x60b000000880,
pattern=0x62900000f200 "t0$\261\"
{|j\336\261\f{\231j||eNJr(\326(\331(((\021((((2\004(||e\017Dpi\336NDr(\326(\003\3
50(((\365((((*(0'(|)(((=((((|)(((=(\005((|||\035(8|)", '(' <repeats 109 times>...,
length=17404, syntax=2339405) at ./regcomp.c:764
#17720 0x00005555558b0a7e in re_compile_pattern (pattern=0x62900000f200
"t0$\261\"
{|j\336\261\f{\231j||eNJr(\326(\331(((\021((((2\004(||e\017Dpi\336NDr(\326(\003\3
50(((\365((((*(0'(|)(((=((((|)(((=(\005((|||\035(8|)", '(' <repeats 109 times>...,
length=17404, bufp=0x60b000000880) at ./regcomp.c:217
#17721 0x000055555585fc4d in make_regexp (s=0x62900000a200 "t0$\261\"
{|j\336\261\f{\231j||eNJr(\326(\331(((\021((((2\004(||e\017Dpi\336NDr(\326(\003\3
50(((\365((((*(0'(|)(((=((((|)(((=(\005((|||\035(8|)", '(' <repeats 109 times>...,
len=17404, ignorecase=false, dfa=true, canfatal=false) at re.c:257
#17722 0x00005555556d2159 in make_regnode (type=Node_regex, exp=0x626000006560)
at /src/Source/gawk_newest/gawk/awkgram.y:5297
#17723 0x00005555556a4257 in yyparse () at
/src/Source/gawk_newest/gawk/awkgram.y:572
#17724 0x00005555556e1ea1 in parse_program (pcode=0x555556351c60 <code_block>,
from_eval=false) at /src/Source/gawk_newest/gawk/awkgram.y:2803
#17725 0x000055555581c88e in main (argc=4, argv=0x7fffffffe3b8) at main.c:504
```

Impact

This vulnerability can be used for causing the crash or long-term loop of the software which would leads to denial of service(DoS). Besides, the attacker can exploit weak points to launch remote code execution.

Reference

POC_FILE