Toybox Bug Analysis

Austin Mordahl

June 22, 2018

Contents

1 Introduction

These bugs were generated by Cppcheck 1.72 and Toybox 0.7.5. Bug reports are classified into the following categories:

True	A bug which exists and 1) its existence is unintended, or 2) whether or not its existence is purposeful is undetermined.
Technically True	A bug for which the content of the cppcheck bug report is true, but whose existence is intended. The difference between a False and Technically True bug report is that the former could theoretically be detected by a more sophisticated implementation of cppcheck.
False	A bug cppcheck finds which, upon further inspection, does not exist in the code. For example, cppcheck indicating a variable is passed to a function without being initialized, when the variable is actually an out parameter and intialized within the function.

2 True Reports

File	blockdev.c
Line	60
Description	Array `cmds[11]` accessed at index 31, which is out of bounds.
Number of Configurations	482

Code Sample

```
52    for (i = 0; i < 32; i++) {
53      long flag = toys.optflags & (1<<i);
54
55      if (!flag) continue;
56
57      if (flag & FLAG_setbsz) val = TT.bsz;
58      else val = !!(flag & FLAG_setro);
59
60      xioctl(fd, cmds[i], &val);
61
62      flag &= FLAG_setbsz|FLAG_setro|FLAG_flushbufs|FLAG_rereadpt|
    FLAG_setrw;
63      if (!flag) printf("%lld\n", (toys.optflags & FLAG_getsz) ? val >>
     9: val);
64    }
```

Status	True
Remarks	`cmd[]` is defined as an integer array of size 11. By using a loop that iterates through the number 31 to access the loop, the program is exceeding the bounds of the array.

File	netstat.c
Line	118
Description	Resource leak: `fp`
Number of Configurations	515

Code Sample

```
111  FILE *fp = fopen(fname, "r");
112
113  if (!fp) {
114     perror_msg("'%s'", fname);
115     return;
116  }
117
118  if(!fgets(toybuf, sizeof(toybuf), fp)) return; //skip header.
```

Status	True
Remarks	`fp` is not closed before the function returns.

File	cmp.c
Line	83
Description	Signed integer overflow for expression `(2147483648)*!(toys.optflags&(1))`.
Number of Configurations	501

Code Sample

```
80 void cmp_main(void)
81 {
82   toys.exitval = 2;
83   loopfiles_rw(toys.optargs, O_CLOEXEC|(WARN_ONLY*!(toys.optflags&
       FLAG_s)), 0,
84     do_cmp);
85 }
```

Status	True
Remarks	The multiplication of the flags will cause integer overflow.

3 Technically True Reports

File	chvt.c
Line	24
Description	Uninitialized variable: `fd`
Number of Configurations	512

Code Sample

```
22 void chvt_main(void)
23 {
24   int vtnum, fd = fd;
25   char *consoles[]={"/dev/console", "/dev/vc/0", "/dev/tty", NULL}, **
       cc;
26
27   vtnum=atoi(*toys.optargs);
28   for (cc = consoles; *cc; cc++)
29     if (-1 != (fd = open(*cc, O_RDWR))) break;
30
31   // These numbers are VT_ACTIVATE and VT_WAITACTIVE from linux/vt.h
32   if (!*cc || fd < 0 || ioctl(fd, 0x5606, vtnum) || ioctl(fd, 0x5607,
       vtnum))
33     perror_exit(0);
34 }
```

Status	Technically True
Remarks	The self-assignment `fd=fd` is likely purposeful, as a method to suppress compiler warnings about an unused variable `fd` before the rest of `chvt_main` was written to use `fd`. However, cppcheck is correct in that `fd`$\bar{\texttt{f}}$`d` is an assignment of the value of an uninitialized variable.

File	date.c
Line	137
Description	Uninitialized variable: `width`
Number of Configurations	511

Code Sample

```
134 static void puts_time(char *fmt, struct tm *tm)
135 {
136   char *s, *snap;
137   long width = width;
138
139   for (s = fmt;;s++) {
140
141     // Find next %N or end
142     if (*(snap = s) == '%') {
143       width = isdigit(*++s) ? *(s++)-'0' : 9;
144       if (*s && *s != 'N') continue;
145     } else if (*s) continue;
146
147     // Don't modify input string if no %N (default format is constant
      string).
148     if (*s) *snap = 0;
149     if (!strftime(toybuf, sizeof(toybuf)-10, fmt, tm))
150       perror_exit("bad format '%s'", fmt);
151     if (*s) {
152       snap = toybuf+strlen(toybuf);
153       sprintf(snap, "%09u", TT.nano);
154       snap[width] = 0;
155     }
156     fputs(toybuf, stdout);
157     if (!*s || !*(fmt = s+1)) break;
158   }
159   xputc('\n');
160 }
```

Status	Technically True
Remarks	See the report for `chvt.c:24`.

File	hwclock.c
Line	89
Description	Uninitialized variable: `s`
Number of Configurations	466

Code Sample

```
88    if (!w) {
89      char *s = s;
90
91      xioctl(fd, RTC_RD_TIME, &tm);
92      if (TT.utc) s = xtzset("UTC0");
93      if ((time = mktime(&tm)) < 0) error_exit("mktime failed");
94      if (TT.utc) {
95        free(xtzset(s));
96        free(s);
97      }
98    }
```

Status	Technically True
Remarks	See the report for `chvt.c:24`.

File	losetup.c
Line	64
Description	Uninitialized variable: `ffd`
Number of Configurations	531

Code Sample

```
63  struct loop_info64 *loop = (void *)(toybuf+32);
64  int lfd = -1, ffd = ffd;
65  unsigned flags = toys.optflags;
66
67  // Open file (ffd) and loop device (lfd)
68
69  if (file) ffd = xopen(file, TT.openflags);
```

Status	Technically True
Remarks	See the report for `chvt.c:24`.

File	switch_root.c
Line	49
Description	Uninitialized variable: `console`
Number of Configurations	486

Code Sample

```
46  char *newroot = *toys.optargs, **cmdline = toys.optargs+1;
47  struct stat st1, st2;
48  struct statfs stfs;
49  int console = console; // gcc's "may be used" warnings are broken.

81  if (TT.console && -1 == (console = open(TT.console, O_RDWR))) {
82    perror_msg("bad console '%s'", TT.console);
83    goto panic;
84  }
```

Status	Technically True
Remarks	See the report for `chvt.c:24`.

File	uudecode.c
Line	29
Description	Uninitialized variable: `m`
Number of Configurations	485

Code Sample

```
29  int ifd = 0, ofd, idx = 0, m = m;
30  char *line = 0, mode[16],
31       *class[] = {"begin%*[ ]%15s%*[ ]%n", "begin-base64%*[ ]%15s%*[
      ]%n"};
32
33  if (toys.optc) ifd = xopenro(*toys.optargs);
34
35  while (!idx) {
36    free(line);
37    if (!(line = get_line(ifd))) error_exit("bad EOF");
38    for (m=0; m < 2; m++) {
39      sscanf(line, class[m], mode, &idx);
40      if (idx) break;
41    }
42  }
```

Status	Technically True
Remarks	See the report for `chvt.c:24`.

File	vmstat.c
Line	51
Description	Uninitialized variable: `name` Uninitialized variable: `p`
Number of Configurations	508

Code Sample

```
51  char *p = p, *name = name;
52  int i, j;
53
54  // We use vmstuff to fill out vmstat_proc as an array of uint64_t:
55  //   Strings starting with / are the file to find next entries in
56  //   Any other string is a key to search for, with decimal value
      right after
57  //   0 means parse another value on same line as last key
58
59  for (i = 0; i<sizeof(vmstuff)/sizeof(char *); i++) {
60    if (!vmstuff[i]) p++;
61    else if (*vmstuff[i] == '/') {
62      xreadfile(name = vmstuff[i], toybuf, sizeof(toybuf));
63
64      continue;
65    } else if (!(p = strafter(toybuf, vmstuff[i]))) goto error;
66    if (1 != sscanf(p, "%"PRIu64"%n", new++, &j)) goto error;
67    p += j;
68  }
69
70  return;
```

Status	Technically True
Remarks	See the report for `chvt.c:24`.

4 False Reports

File	lsm.h
Line	63
Description	Uninitialized variable: result
Number of Configurations	432[1]

Code Sample

```
55 static inline char *lsm_context(void)
56 {
57   int ok = 0;
58   char *result;
59
60   if (CFG_TOYBOX_SMACK) ok = smack_new_label_from_self(&result) > 0;
61   else ok = getcon(&result) == 0;
62
63   return ok ? result : strdup("?");
64 }
```

Status	False
Remarks	In configurations including `TOYBOX_SMACK` and `TOYBOX_SELINUX` `smack_new_label_from_self` and `getcon` are replaced with the value -1, respectively. In other configurations, `*result` is an out parameter.

[1]The actual cppcheck bug reports listed various C source code files which included this header as the source of the bug, even though `lsm.h` was the actual source. This is the number of total occurrences of the bug across multiple files.

File	base64.c
Line	35
Description	Expression ‘`this.base64.columns&&++*x == this.base64.columns`’ depends on order of evaluation of side effects.
Number of Configurations	478

Code Sample

```
31 static void wraputchar(int c, int *x)
32 {
33   putchar(c);
34   TT.total++;
35   if (TT.columns && ++*x == TT.columns) {
36     *x = 0;
37     xputc('\n');
38   };
39 }
```

Status	False
Remarks	Although `TT.columns` appears twice in the same expression, it is modified neither time. Thus, the order of evaluation of side effects does not matter.

File	tail.c
Line	188
Description	Memory is allocated but not initialized: `try`
Number of Configurations	655

Code Sample

```
181         int len = new->len, count;
182         char *try = new->data;
183 
184         // First character _after_ a newline starts a new line, which
185         // works even if file doesn't end with a newline
186         for (count=0; count<len; count++) {
187           if (linepop) lines++;
188           linepop = try[count] == '\n';
```

Status	False
Remarks	The `for` loop causing cppcheck to give a warning is actually only testing `try[count]` for equality.

File	args.c
Line	309
Description	Uninitialized variable: temp
Number of Configurations	519

Code Sample

```
301     else if (-1 != (idx = stridx("<>=", *options))) {
302       if (new->type == '#') {
303         long l = strtol(++options, &temp, 10);
304         if (temp != options) new->val[idx].l = l;
305       } else if (CFG_TOYBOX_FLOAT && new->type == '.') {
306         FLOAT f = strtod(++options, &temp);
307         if (temp != options) new->val[idx].f = f;
308       } else if (CFG_TOYBOX_DEBUG) error_exit("<>= only after .#");
309       options = --temp;
```

Status	False
Remarks	`temp` is necessarily initialized by either a call to `strol` or `strod`. In other words, `temp` will necessarily be initialized.