Toybox Bug Analysis

Austin Mordahl

June 21, 2018

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

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.
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.
True	A bug which exists and 1) its existence is unintended, or 2) whether or not its existence is purposeful is undetermined.

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

Code Sample

```
void blockdev_main(void)
{
  int cmds[] = {BLKRRPART, BLKFLSBUF, BLKGETSIZE64,
    BLKGETSIZE, BLKGETSIZE64, BLKBSZSET,
    BLKBSZGET, BLKSSZGET, BLKROGET,
    BLKROSET, BLKROSET};
  char **ss;
  long long val = 0;

  if (!toys.optflags) help_exit("need --option");

  for (ss = toys.optargs;  *ss; ss++) {
    int fd = xopenro(*ss), i;

    // Command line order discarded so perform
    //  multiple operations in flag order
    for (i = 0; i < 32; i++) {
      long flag = toys.optflags & (1<<i);

      if (!flag) continue;

      if (flag & FLAG_setbsz) val = TT.bsz;
      else val = !!(flag & FLAG_setro);

      xioctl(fd, cmds[i], &val);

      flag &= FLAG_setbsz|FLAG_setro|FLAG_flushbufs|
  FLAG_rereadpt|FLAG_setrw;
      if (!flag) printf("%lld\n", (toys.optflags & FLAG_getsz) ?
      val >> 9: val);
    }
    xclose(fd);
  }
}
```

Status	True[1]
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.

[1] This seems suspiciously obvious; I need to run more tests to determine whether this is correct under some binary magic the program is doing.

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

Code Sample

```c
static void show_ip(char *fname)
{
  char *ss_state = "UNKNOWN", buf[12], *s, *label = strrchr(fname, '/')+1;
  char *state_label[] = {"", "ESTABLISHED", "SYN_SENT", "SYN_RECV", "FIN_WAIT1",
                         "FIN_WAIT2", "TIME_WAIT", "CLOSE", "CLOSE_WAIT",
                         "LAST_ACK", "LISTEN", "CLOSING", "UNKNOWN"};
  struct passwd *pw;
  FILE *fp = fopen(fname, "r");

  if (!fp) {
     perror_msg("'%s'", fname);
     return;
  }

  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

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

Status	True (further study required)
Remarks	The multiplication of the flags will cause integer overflow. Whether or not this behavior is intended will require further investigation.

File	lsm.h
Line	63
Description	Uninitialized variable: result
Number of Configurations	432[2]

Code Sample

```
static inline char *lsm_context(void)
{
  int ok = 0;
  char *result;

  if (CFG_TOYBOX_SMACK) ok = smack_new_label_from_self(&result) > 0;
  else ok = getcon(&result) == 0;

  return ok ? result : strdup("?");
}
```

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.

[2] 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

```
static void wraputchar(int c, int *x)
{
  putchar(c);
  TT.total++;
  if (TT.columns && ++*x == TT.columns) {
    *x = 0;
    xputc('\n');
  };
}
```

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

Code Sample

```
void chvt_main(void)
{
  int vtnum, fd = fd;
  char *consoles[]={"/dev/console", "/dev/vc/0",
        "/dev/tty", NULL}, **cc;

  vtnum=atoi(*toys.optargs);
  for (cc = consoles; *cc; cc++)
    if (-1 != (fd = open(*cc, O_RDWR))) break;

  // These numbers are VT_ACTIVATE and VT_WAITACTIVE from linux/vt.h
  if (!*cc || fd < 0 || ioctl(fd, 0x5606, vtnum) ||
      ioctl(fd, 0x5607, vtnum))
    perror_exit(0);
}
```

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 fdf̄d 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

```
static void puts_time(char *fmt, struct tm *tm)
{
  char *s, *snap;
  long width = width;

  for (s = fmt;;s++) {

    // Find next %N or end
    if (*(snap = s) == '%') {
      width = isdigit(*++s) ? *(s++)-'0' : 9;
      if (*s && *s != 'N') continue;
    } else if (*s) continue;

    // Don't modify input string if
    //  no %N (default format is constant string).
    if (*s) *snap = 0;
    if (!strftime(toybuf, sizeof(toybuf)-10, fmt, tm))
      perror_exit("bad format '%s'", fmt);
    if (*s) {
      snap = toybuf+strlen(toybuf);
      sprintf(snap, "%09u", TT.nano);
      snap[width] = 0;
    }
    fputs(toybuf, stdout);
    if (!*s || !*(fmt = s+1)) break;
  }
  xputc('\n');
}
```

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

```
if (!w) {
  char *s = s;

  xioctl(fd, RTC_RD_TIME, &tm);
  if (TT.utc) s = xtzset("UTC0");
  if ((time = mktime(&tm)) < 0) error_exit("mktime failed");
  if (TT.utc) {
    free(xtzset(s));
    free(s);
  }
}
```

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

```
static void loopback_setup(char *device, char *file)
{
  struct loop_info64 *loop = (void *)(toybuf+32);
  int lfd = -1, ffd = ffd;
  unsigned flags = toys.optflags;

  // Open file (ffd) and loop device (lfd)

  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

```
void switch_root_main(void)
{
  char *newroot = *toys.optargs, **cmdline = toys.optargs+1;
  struct stat st1, st2;
  struct statfs stfs;
  int console = console; // gcc's "may be used" warnings are broken.

  // ...

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

  // ...

  if (TT.console) {
    int i;
    for (i=0; i<3; i++) if (console != i) dup2(console, i);
    if (console>2) close(console);
  }
  execv(*cmdline, cmdline);
  perror_msg("Failed to exec '%s'", *cmdline);
panic:
  if (toys.optflags & FLAG_h) for (;;) wait(NULL);
}
```

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

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

Code Sample

```
static void do_tail(int fd, char *name)
{
  // ...

  if (bytes<0 || lines<0) {
    struct line_list *list = 0, *new;

    // The slow codepath is always needed, and can handle all input,
    // so make lseek support optional.
    if (CFG_TAIL_SEEK && try_lseek(fd, bytes, lines)) return;

    // Read data until we run out, keep a trailing buffer
    for (;;) {
      // Read next page of data, appending to linked list in order
      if (!(new = get_chunk(fd, sizeof(toybuf)))) break;
      dlist_add_nomalloc((void *)&list, (void *)new);

      // If tracing bytes, add until we have enough, discarding
    overflow.
      if (TT.bytes) {
        bytes += new->len;
        if (bytes > 0) {
          while (list->len <= bytes) {
            bytes -= list->len;
            free(dlist_pop(&list));
          }
          list->data += bytes;
          list->len -= bytes;
          bytes = 0;
        }
      } else {
        int len = new->len, count;
        char *try = new->data;

        // First character _after_ a newline starts a new line, which
        // works even if file doesn’t end with a newline
        for (count=0; count<len; count++) {
          if (linepop) lines++;
          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	uudecode.c
Line	29
Description	Uninitialized variable: `m`
Number of Configurations	485

Code Sample

```
void uudecode_main(void)
{
  int ifd = 0, ofd, idx = 0, m = m;
  char *line = 0, mode[16],
       *class[] = {"begin%*[ ]%15s%*[ ]%n", "begin-base64%*[ ]%15s%*[
    ]%n"};

  // ...
}
```

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

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

Code Sample

```c
static void get_vmstat_proc(struct vmstat_proc *vmstat_proc)
{
  char *vmstuff[] = { "/proc/stat", "cpu ", 0, 0, 0, 0, 0, 0,
    "intr ", "ctxt ", "procs_running ", "procs_blocked ", "/proc/
    meminfo",
    "MemFree: ", "Buffers: ", "Cached: ", "SwapFree: ", "SwapTotal: ",
    "/proc/vmstat", "pgpgin ", "pgpgout ", "pswpin ", "pswpout " };
  uint64_t *new = (uint64_t *)vmstat_proc;
  char *p = p, *name = name;
  int i, j;

  // ...
}
```

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