

Toybox Bug Analysis

Austin Mordahl

October 8, 2018

Contents

1	Introduction	1
2	True Reports	2
3	Technically True Reports	5
4	False Reports	12

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 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 the tool. Often, these bugs are “true but uninteresting.”
False	A bug the checker finds which, upon further inspection, does not exist in the code. For example, the checker indicating a variable is passed to a function without being initialized, when the variable is actually an out parameter and initialized within the function.

2 True Reports

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

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

Status	True
Remarks	<code>cmd[]</code> 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.
Features	CONFIG_BLOCKDEV

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.
Features	CONFIG_NETSTAT

	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&
84         FLAG_s)), 0,
85         do_cmp);
86 }

```

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

3 Technically True Reports

	File	chvt.c
	Line	24
	Description	Uninitialized variable: fd
	Number of Configurations	512
Code Sample		
<pre> 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 }</pre>		
Status	Technically True	
Remarks	The self-assignment <code>fd=fd</code> is likely purposeful, as a method to suppress compiler warnings about an unused variable <code>fd</code> before the rest of <code>chvt_main</code> was written to use <code>fd</code> . However, <code>cppcheck</code> is correct in that <code>fdfd</code> is an assignment of the value of an uninitialized variable.	
Feature	CONFIG_CHVT	

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
148         // string).
149         if (*s) *snap = 0;
150         if (!strftime(toybuf, sizeof(toybuf)-10, fmt, tm))
151             perror_exit("bad format '%s'", fmt);
152         if (*s) {
153             snap = toybuf+strlen(toybuf);
154             sprintf(snap, "%09u", TT.nano);
155             snap[width] = 0;
156         }
157         fputs(toybuf, stdout);
158         if (!*s || !(fmt = s+1)) break;
159     }
160     xputc('\n');

```

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

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.utco) s = xtzset("UTC0");
93         if ((time = mktime(&tm)) < 0) error_exit("mktime failed");
94         if (TT.utco) {
95             free(xtzset(s));
96             free(s);
97         }
98     }

```

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

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

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

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

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

Code Sample	
46	<code>char *newroot = *toys.optargs, **cmdline = toys.optargs+1;</code>
47	<code>struct stat st1, st2;</code>
48	<code>struct statfs stfs;</code>
49	<code>int console = console; // gcc's "may be used" warnings are broken.</code>
81	<code>if (TT.console && -1 == (console = open(TT.console, O_RDWR))) {</code>
82	<code> perror_msg("bad console '%s'", TT.console);</code>
83	<code> goto panic;</code>
84	<code>}</code>

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

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

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

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

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

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

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

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 <code>smack_new_label_from_self</code> and <code>getcon</code> are replaced with the value -1, respectively. In other configurations, <code>*result</code> is an out parameter.

¹The 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 <code>'this.base64.columns&&++*x == this.base64.columns'</code> depends on order of evaluation of side effects.
Number of Configurations	478

Code Sample

```

31 static void wrapputchar(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 <code>TT.columns</code> 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: <code>try</code>
	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 <code>try[count]</code> for equality.

	File	args.c
	Line	309
	Description	Uninitialized variable: temp
	Number of Configurations	519
Code Sample		
301	<pre> else if (-1 != (idx = stridx("<=>", *options))) { if (new->type == '#') { long l = strtol(++options, &temp, 10); if (temp != options) new->val[idx].l = l; } else if (CFG_TOYBOX_FLOAT && new->type == '.') { FLOAT f = strtod(++options, &temp); if (temp != options) new->val[idx].f = f; } else if (CFG_TOYBOX_DEBUG) error_exit("<=> only after .#"); options = --temp; </pre>	
	Status	False
	Remarks	temp is necessarily initialized by either a call to strtol or strtod.

File	lib.c
Line	975
Description	Buffer is accessed out of bounds: ``xwr``
Number of Configurations	986

Code Sample

```

965 char c, d;
966 int i, bit;
967
968 buf[10]=0;
969 for (i=0; i<9; i++) {
970     bit = mode & (1<<i);
971     c = i%3;
972     if (!c && (mode & (1<<((d=i/3)+9)))) {
973         c = "tss"[d];
974         if (!bit) c &= ~0x20;
975     } else c = bit ? "xwr"[c] : '-';
976     buf[9-i] = c;
977 }

```

Status	False
Remarks	``xwr`` will never be accessed out of bounds. c is assigned by the expression <code>c = i % 3</code> , which will give c the value of 0, 1, or 2.