Toybox Bug Analysis – infer

Austin Mordahl

July 16, 2018

1 Introduction

These bugs were generated by Infer v0.15.0 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 intialized within the function.

2 True Reports

File	uptime.c
Line	54
Bug Type	NULL_DEREFERENCE
Description	pointer tm last assigned on line 53 could be null and is dereferenced at line 54, column 34
Number of Configurations	495

Code Sample

```
53  tm = localtime(&t);
54  xprintf(" %02d:%02d:%02d up ", tm->tm_hour, tm->tm_min, tm->tm_sec);
```

Status	True
Remarks	localtime may fail and return a null pointer, causing tm to be null.

File	lib.c
Line	1268
Bug Class	MEMORY_LEAK
Description	gr is not reachable after line 1268, column 3.
Number of Configurations	986

Code Sample

```
1264 // Return group name or string representation of number, returned buffer
1265 // lasts until next call.
1266 char *getgroupname(gid_t gid)
1267 {
1268   struct group *gr = bufgetgrgid(gid);
1269   static char gnum[12];
1270 
1271   sprintf(gnum, "%u", (unsigned)gid);
1272   return gr ? gr->gr_name : gnum;
1273 }
```

Status	True
Remarks	The call to `bufgetgrgid` initializes overwrites the `list` pointer with pointer to new memory without freeing the memory it pointed to initially. Although the buffer is meant to last past the end of the function, the memory is orphaned each time `list` is reassigned.
Similar Bugs	`lib.c:1257` – pw is not reachable after line 1257, column 3.

File	lib.c
Line	1268
Bug Class	MEMORY_LEAK
Description	pw is not reachable after line 1257, column 3.
Number of Configurations	986

Code Sample

```
1255 char *getusername(uid_t uid)
1256 {
1257   struct passwd *pw = bufgetpwuid(uid);
1258   static char unum[12];
1259 
1260   sprintf(unum, "%u", (unsigned)uid);
1261   return pw ? pw->pw_name : unum;
1262 }
```

Status	True
Remarks	The same explanation for lib.c:1264 applies here.

File	mkflags.c
Line	126
Bug Type	`NULL_DEREFERENCE`
Description	pointer `new` last assigned on line 124 could be null and is dereferenced at line 126, column 7.
Number of Configurations	986

Code Sample

```
124     struct flag *new = calloc(sizeof(struct flag), 1);
125
126     new->command = string++;
127     new->next = list;
128     list = new;
```

Status	True
Remarks	`calloc` *can* fail, leaving `new` as a null pointer. Although this is true, it's not particularly interesting, as `calloc` very rarely fails.

File	mkflags.c
Line	101
Bug Type	`NULL_DEREFERENCE`
Description	pointer `new` last assigned on line 124 could be null and is dereferenced at line 126, column 7.
Number of Configurations	986

Code Sample

```
124     struct flag *new = calloc(sizeof(struct flag), 1);
125
126     new->command = string++;
127     new->next = list;
128     list = new;
```

Status	True
Remarks	`calloc` can fail, leaving `new` as a null pointer.

3 Technically True Reports

File	nohup.c
Line	27
Bug Type	RESOURCE_LEAK
Description	resource acquired by call to `open()` at line 27, column 15 is not released after line 27, column 9.
Number of Configurations	476

Code Sample

```
if (isatty(1)) {
  close(1);
  if (-1 == open("nohup.out", O_CREAT|O_APPEND|O_WRONLY,
      S_IRUSR|S_IWUSR ))
  {
    char *temp = getenv("HOME");

    temp = xmprintf("%s/%s", temp ? temp : "", "nohup.out");
    xcreate(temp, O_CREAT|O_APPEND|O_WRONLY, 0600);
    free(temp);
  }
}
```

Status	Technically True
Remarks	The file descriptor opened cannot be closed, as the whole point is to redirect `stdin` and `stdout` and then run a command elsewhere. This is not a bug, but at the same time infer is not wrong that there is an open file descriptor that is not being closed. Other bugs which involve open resources that are not closed but are meant to stay open are listed below.
Similar Bugs	`oneit.c:72` – resource acquired by call to `xopen_stdio()` at line 72, column 12 is not released after line 72, column 12.

4 False Reports

File	grep.c
Line	184
Bug Type	UNINITIALIZED_VALUE
Description	The value read from `matches.rm_so` was never initialized.
Number of Configurations	507

Code Sample

```
178     if (toys.optflags & FLAG_v) {
179       if (toys.optflags & FLAG_o) {
180         if (rc) skip = matches.rm_eo = strlen(start);
181         else if (!matches.rm_so) {
182           start += skip;
183           continue;
184         } else matches.rm_eo = matches.rm_so;
185       } else {
186         if (!rc) break;
187         matches.rm_eo = strlen(start);
188       }
189       matches.rm_so = 0;
190     } else if (rc) break;
```

Status	False
Remarks	Were `matches.rm_so` a singular variable, infer would be correct, becuase the initialization of `matches.rm_so` would be out of scope. However, `matches` is a struct which is in scope. Additionally, the else if clause checks whether `matches.rm_so` exists; line 184 will not be reached if `matches.rm_so` is not initialized.

File	xwrap.c
Line	389
Bug Type	RESOURCE_LEAK
Description	resource acquired by call to xopen_stdio() at line 389, column 19 is not released after line 389, column 3.
Number of Configurations	986

Code Sample

```c
int xcreate_stdio(char *path, int flags, int mode)
{
  int fd = open(path, (flags^O_CLOEXEC)&~WARN_ONLY, mode);

  if (fd == -1) ((mode&WARN_ONLY) ? perror_msg_raw : perror_exit_raw)(path);
  return fd;
}

// Die unless we can open a file, returning file descriptor.
int xopen_stdio(char *path, int flags)
{
  return xcreate_stdio(path, flags, 0);
}
```

Status	False
Remarks	xopen_stdio() automatically closes a file unless the O_CLOEXEC flag is passed to it (behaves opposite other functions which open files). There are two calls to xopen_stdio() which do not pass O_CLOEXEC. The first is in oneit.c, line 99. Here, the file descriptors are kept open on purpose, redirecting stdin, stdout, and stderr. The same pattern is used in the second occurrence. in getty.c.

File	xwrap.c
Line	458
Bug Type	`NULL_DEREFERENCE`
Description	pointer `null` could be null and is dereferenced by call to `getcwd()` at line 458, column 15.
Number of Configurations	986

Code Sample

```
456 char *xgetcwd(void)
457 {
458   char *buf = getcwd(NULL, 0);
459   if (!buf) perror_exit("xgetcwd");
460
461   return buf;
462 }
```

Status	False
Remarks	The usage of the `NULL` pointer should not trigger this error.

File	xwrap.c
Line	264
Bug Type	`UNINITIALIZED_VALUE`
Description	The value read from `cestnepasun[_]` was never initialized.
Number of Configurations	986

Code Sample

```
if (pipes) {
  if (pipes[0] != -1) close(cestnepasun[0]);
  if (pipes[1] != -1) close(cestnepasun[3]);
}
```

Status	False
Remarks	`cestnepasun` is initialized by a call to `pipe`. This is representative of a broader class of bugs falling into the `UNINTIALIZED_VALUE` type. These bugs take the form of two separate branches with the same condition, wherein a variable is intialized in the first and used in the second (i.e. `int a; if (b) a = 5; ...; if (b) int c = a;` Other false bugs which fall into this class are listed below.
Similar Bugs	`xwrap.c:287` – The value read from `pipe` was never initialized. `mount.c:314` – The value read from `mtl` was never initialized. `comm.c:79` – The value read from `file[_]` was never intiialized. `comm.c:68` – The value read from `file[_]` was never intiialized. `kill.c:145` – The value read from `signum` was never intialized. `xwrap.c:232` – The value read from `cestnepasun[_]` was never initialized. `xwrap.c:213` – The value read from `cestnepasun[_]` was never initialized. `xwrap.c:790` – The value read from `d` was never intialized. `xwrap.c:737` – The value read from `fd` was never initialized. `comm.c:62` – The value read from `file[_]` was never initialized. `xwrap.c:225` – The value read from `cestnepasun[_]` was never initialized.

File	ulimit.c
Line	95
Bug Type	`UNINITIALIZED_VALUE`
Description	The value read from `rr.rlim_cur` was never initialized.
Number of Configurations	510

Code Sample

```
84        if ((1<<i)&FLAG_p) {
85          if (toys.optflags&FLAG_H)
86            xreadfile("/proc/sys/fs/pipe-max-size", toybuf, sizeof(
    toybuf));
87          else {
88            int pp[2];
89
90            xpipe(pp);
91            sprintf(toybuf, "%d\n", fcntl(*pp, F_GETPIPE_SZ));
92          }
93          printf("%s", toybuf);
94        } else {
95          rlim_t rl = (toys.optflags&FLAG_H) ? rr.rlim_max : rr.
    rlim_cur;
96
97          if (rl == RLIM_INFINITY) printf("unlimited\n");
98          else printf("%ld\n", (long)rl);
99        }
```

Status	False
Remarks	`rr` is initialized by a call to `prlimit`. This bug is representative of a broader class of bugs falling into the `UNINITIALIZED_VALUE` type. These bugs take the form of a struct being initialized and then a field in that struct being referenced later. Other false bugs which fall into this class are listed below.
Similar Bugs	`od.c:88` – The value read from `fdl.ld` was never initialized. `od.c:85` – The value read from `fdl.d` was never initialized.

File	ls.c
Line	428
Bug Type	`UNINITIALIZED_VALUE`
Description	The value read from `totals[_]` was never initialized.
Number of Configurations	655

Code Sample

```
  if (flags & FLAG_s) {
    print_with_h(tmp, st->st_blocks, 512);
    printf("%*s ", totals[6], tmp);
  }
```

Status	False
Remarks	`totals` was initialized by memset. Other bugs in which the variable was initialized by memset, memcpy, or any other function which accepts out parameters are listed below.
Similar Bugs	`md5sum.c:157` – The value read from `x[_]` was never initialized.
Similar Bugs	`ps.c:1453` – The value read from `run[_]` was never initialized.
	`md5sum.c:143` – The value read from `x[_]` was never initialized.
	`ftpget.c:150` – The value read from `port` was never initialized.
	`ftpget.c:151` – The value read from `si6.sin6_family` was never initialized.
	`md5sum.c:147` – The value read from `x[_]` was never initialized.
	`ls.c:369` – The value read from `dtlen` was never intialized.

File	pmap.c
Line	86
Bug Type	`UNINITIALIZED_VALUE`
Description	The value read from swap was never intialized.
Number of Configurations	500

Code Sample

```
        if (0<sscanf(line, "Pss: %lld", &pss)
            || 0<sscanf(line, "Private_Dirty: %lld", &dirty)
            || 0<sscanf(line, "Swap: %lld", &swap)) xx++;
        free(line);
        if (xx<3) continue;
        line = oldline;
        name = basename(name);
        xx = 0;
        printf("% 7lld %7lld %7lld ", pss, dirty, swap);
```

Status	TBD
Remarks	