Shape Analysis Using Separation Logic

Author
Tomáš Brablec

Supervisors
Tomáš Dacík, Tomáš Vojnar

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Motivation

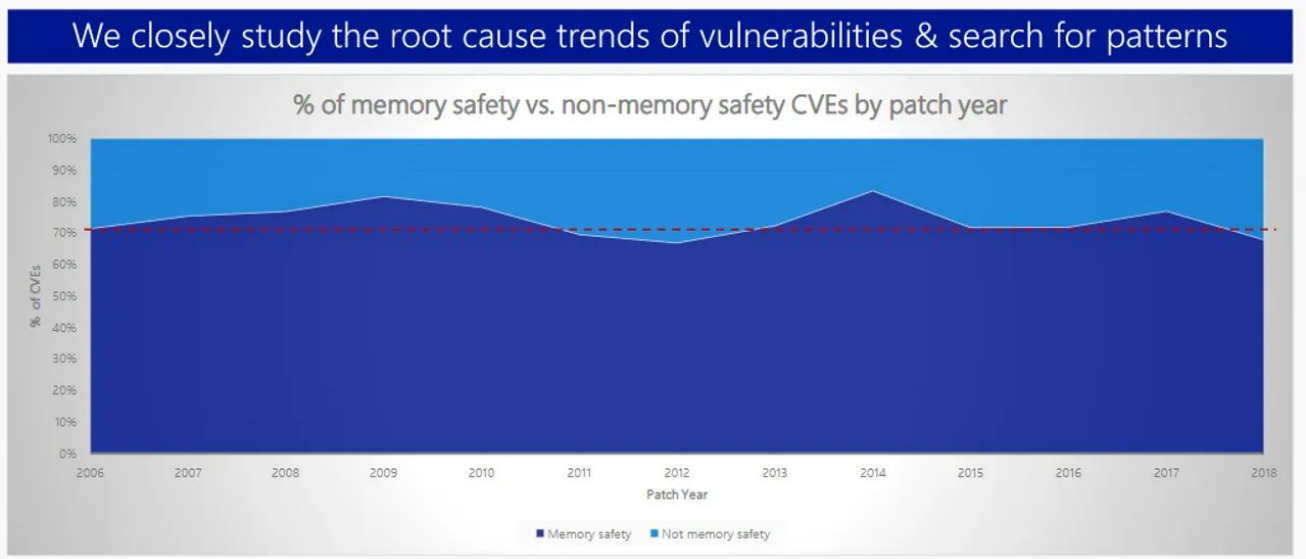
- Manual memory management creates a whole class of bugs
- use-after-free, double-free, memory leaks, …
- Memory errors are a common source of security vulnerabilities
- Linked lists are a common data structure in low level software

CVE-2024-12382 Detail

Description

Use after free in Translate in Google Chrome prior to 131.0.6778.139 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page. (Chromium security severity: High)

Example of a recent memory safety bug



Microsoft: Around 70% of CVEs are memory related

Verification Tool

- The goal is to create a tool for verification of programs
- Abstract program states are encoded in separation logic
- Solver (Astral) used to check satisfiability
- By checking satisfiability, the tool proves properties of the program
- The secondary goal is to test Astral itself in real-world use-cases

Astral

Software Analyzers

Current State

- Implementation mostly finished (AST preprocessing, abstraction, simplification of formulas, function summaries, GUI output, …)
- Testing and optimization is in progress
- Benchmarking on public datasets of programs (SV-COMP) remains to be done
- Analysis is implemented for most common types of lists (singly/doubly linked lists, nested lists)

Results in Ivette (Frama-C GUI)

Ivette

File Edit View Window Help

Source Code* | Console | Properties | declaration

Source Code — testmulti.c

```
10          struct SLL *next;
11          int data;
12  } NL;
13
14  SLL *construct_sll() {
15      SLL *start = calloc(1, 1);
16      if (start == NULL) {
17          return NULL;
18      }
19
20      SLL *s = start;
21      int nondeterministic;
22      while (nondeterministic) {
23          s->next = calloc(1, 1);
24          if (s->next == NULL) {
25              break;
26          }
27
28          s = s->next;
29      }
30
31      s->next = NULL;
32      return start;
```

AST

```
      }
    }
    s_9 = start_8;
    while (_const) {
      {
        struct SLL *s_9_1;
        s_9_1 = (struct SLL *)calloc(_const,_const);
        s_9->next = s_9_1;
      }
      {
        struct SLL *s_9_5;
        s_9_5 = s_9->next;
        if (s_9_5 == _nil) {
          break;
        }
      }
      s_9 = s_9->next;
    }
    s_9->next = _nil;
    __retres_7 = start_8;
    return_label: return __retres_7;
  }
```

Messages 382 / 384

K	E	Categ...	Message	Decl...	Location
?	sl	do_inst:	previous state: (start != nil) * (start -> t:s_4,n:fresh!1) * (ls_0+(...	constru	testmulti.c:4
?	sl	do_edge	original state: (s_4_3 -> t:nil,n:nil) * (start != nil) * (start -> t:s_...	constru	testmulti.c:4
?	sl	combine_	old state: (nil = s_4_3) * (start = s_4) * (start != nil) * (start_2!4...	constru	testmulti.c:4
?	sl	do_inst:	previous state: (start != nil) * (start -> t:s_4,n:fresh!1) * (s_4_3...	constru	testmulti.c:4
!	sl		leak of atom nls_0+(s_4!12,nil,nil)	constru	testmulti.c:4
!	sl		leak of atom nls_0+(s_4!14,nil,nil)	constru	testmulti.c:4
?	sl	do_edge	original state: (s_4 -> t:s_4_3,n:fresh!4) * (nls_0+(s_4!12,nil,ni...	constru	testmulti.c:4
?	sl	combine_	old state: (start = s_4) * (s_4 != nil) * (start_2!4 != nil) * (s_4 ->...	constru	testmulti.c:4
?	sl	do_edge	original state: (start != nil) * (start -> t:s_4,n:fresh!1) * (ls_0+(fr...	constru	testmulti.c:4
?	sl	combine	old state: (start = s_4) * (s_4 != nil) * (start_2!4 != nil) * (s_4 ->...	constru	testmulti.c:4

```
input formula:
(list != nil) * (nls_2+(list,nil,nil))
reachable subformula:
(s = A_s) * (nls_2+(s,nil,nil)) * (s != nil)
unreachable subformula:
(list != nil)
```

Current functior

Search

Category

Message

Kind

- Result
- Feedback
- Debug
- Warning
- Error
- Failure

Emitter

- Kernel
- Aorai
- Cg
- Dive
- E-acsl
- Eva
- From

Inspector

Stmt `s_9 = s_9->next;`

Location `testmulti.c:28`

state
```
(ls_1+(start_8,s_9)) * (start_8 != nil) * (ls_2+(s_9,nil))
(start_8 != nil) * (start_8 -> s_9) * (ls_2+(s_9,nil))
(start_8 = s_9) * (s_9 != nil) * (ls_2+(s_9,nil))
```

Stmt `s_9_5 = s_9->next;`

Location `testmulti.c:24`

state
```
(ls_1+(start_8,s_9)) * (start_8 != nil) * (ls_1+(s_9,nil))
(start_8 != nil) * (start_8 -> s_9) * (ls_1+(s_9,nil))
(start_8 = s_9) * (s_9 != nil) * (ls_1+(s_9,nil))
```

ON | Messages | Console

Conclusion

- Implemented a verification tool able to analyze programs working with linked lists
- Discovered several bugs in Astral leading to incorrect results during debugging

Possible extensions

- Analysis of other types of lists
- Under-approximating analysis mode (useful for finding bugs)