jqlang / jq Public

Code Issues 358 Pull requests 88 Discussions Actions Wiki

Commit fb59f14

itchyny committed 3 days ago · 29 / 30

```
Limit path depth to prevent stack overflow

Deeply nested path arrays can cause unbounded recursion in
`jv_setpath`, `jv_getpath`, and `jv_delpaths`, leading to
stack overflow. Add a depth limit of 10000 to match the
existing `tojson` depth limit. This fixes CVE-2026-33947.
```

master

1 parent 2f09060 commit fb59f14

2 files changed +46 -0 lines changed

Top

Filter files...

- src
  - jv_aux.c
- tests
  - jq.test

2 files changed +46 -0 lines changed

Search within code

src/jv_aux.c

```
@@ -365,13 +365,23 @@ static jv jv_dels(jv t, jv keys) {
365 365   return t;
366 366 }
367 367
    368 + #ifndef MAX_PATH_DEPTH
    369 + #define MAX_PATH_DEPTH (10000)
    370 + #endif
    371 +
```

```
368  372    jv jv_setpath(jv root, jv path, jv value) {
369  373      if (jv_get_kind(path) != JV_KIND_ARRAY) {
370  374        jv_free(value);
371  375        jv_free(root);
372  376        jv_free(path);
373  377        return jv_invalid_with_msg(jv_string("Path must be specified as an
                array"));
374  378      }
     379  +   if (jv_array_length(jv_copy(path)) > MAX_PATH_DEPTH) {
     380  +     jv_free(value);
     381  +     jv_free(root);
     382  +     jv_free(path);
     383  +     return jv_invalid_with_msg(jv_string("Path too deep"));
     384  +   }
375  385      if (!jv_is_valid(root)){
376  386        jv_free(value);
377  387        jv_free(path);
     ↓ ↑     @@ -424,6 +434,11 @@ jv jv_getpath(jv root, jv path) {
424  434        jv_free(path);
425  435        return jv_invalid_with_msg(jv_string("Path must be specified as an
                array"));
426  436      }
     437  +   if (jv_array_length(jv_copy(path)) > MAX_PATH_DEPTH) {
     438  +     jv_free(root);
     439  +     jv_free(path);
     440  +     return jv_invalid_with_msg(jv_string("Path too deep"));
     441  +   }
427  442      if (!jv_is_valid(root)) {
428  443        jv_free(path);
429  444        return root;
     ↓ ↑     @@ -502,6 +517,12 @@ jv jv_delpaths(jv object, jv paths) {
502  517          jv_free(elem);
503  518          return err;
504  519        }
     520  +     if (jv_array_length(jv_copy(elem)) > MAX_PATH_DEPTH) {
     521  +       jv_free(object);
     522  +       jv_free(paths);
     523  +       jv_free(elem);
```

```
524 +       return jv_invalid_with_msg(jv_string("Path too deep"));
525 +     }
505 526       jv_free(elem);
506 527     }
507 528     if (jv_array_length(jv_copy(paths)) == 0) {
```

tests/jq.test

```
@@ -2568,3 +2568,28 @@ true
2568 2568 reduce range(10001) as $_ ([];[.]) | tojson | contains("<skipped: too deep>")
2569 2569 null
2570 2570 true
2571 +
2572 + # regression test for CVE-2026-33947
2573 + setpath([range(10000) | 0]; 0) | flatten
2574 + null
2575 + [0]
2576 +
2577 + try setpath([range(10001) | 0]; 0) catch .
2578 + null
2579 + "Path too deep"
2580 +
2581 + getpath([range(10000) | 0])
2582 + null
2583 + null
2584 +
2585 + try getpath([range(10001) | 0]) catch .
2586 + null
2587 + "Path too deep"
2588 +
2589 + delpaths([[range(10000) | 0]])
2590 + null
2591 + null
2592 +
2593 + try delpaths([[range(10001) | 0]]) catch .
2594 + null
2595 + "Path too deep"
```

Comments 0

Please sign in to comment.