New issue Jump to bottom

Ill-formed oneof message leads to calling free on an arbitrary pointer #647

Closed var-const opened this issue on Mar 19, 2021 · 3 comments

Labels Component-Decoder FixedInGit Priority-High Type-Defect

var-const commented on Mar 19, 2021 • edited

This affects `0.3.9.7` (and probably lower versions, though I haven't checked) with `PB_ENABLE_MALLOC` enabled.

Specially crafted bytes can make `pb_decode` eventually call `pb_free` on an arbitrary pointer. Here's the smallest repro case I could make:

```
/*
// Equivalent to:

syntax = "proto3";
message Repro {
  oneof value_type {
    bool boolean_value = 1;
    string bytes_value = 5;
  }
}
*/

typedef struct _Repro {
    pb_size_t which_value_type;
    union {
        bool boolean_value;
        pb_bytes_array_t *bytes_value;
    };
} Repro;

const pb_field_t Repro_fields[] = {
    PB_ANONYMOUS_ONEOF_FIELD(value_type,   1, BOOL    , ONEOF, STATIC  , FIRST, Repro, boolean_value, boolean_value, 0),
    PB_ANONYMOUS_ONEOF_FIELD(value_type,   5, BYTES   , ONEOF, POINTER , UNION, Repro, bytes_value, bytes_value, 0),
    PB_LAST_FIELD
};

int main() {
  const uint8_t bytes[] = {0x08, 0x08, 0x2d};
  size_t size = 3;

  pb_istream_t stream = pb_istream_from_buffer(bytes, size);
  Repro repro{};
  pb_decode(&stream, Repro_fields, &repro);
}
```

Running this leads to:

```
malloc: *** error for object 0x1: pointer being freed was not allocated
```

I can repro this on both Linux and Mac.

What I believe happens is:

- the first two bytes are interpreted as `boolean_value` and result in `iter.pData` being set to `1`;
- the third byte is interpreted as a field tag referring to `bytes_value`. Because there are no more bytes in the input, decoding the field fails (with `end-of-stream`). However, the current field is reset to `5` from `1` while `iter.pData` is not cleared and is still set to `1`;
- seeing that decoding failed, `pb_decode` tries to release the message. `pb_release_single_field`, thinking that the current field is `5`, considers the contents of `iter.pData` to refer to a dynamically-allocated array and calls `pb_free` on it.

I'm not sure what the right fix would be -- perhaps `iter.pData` should be set to null when oneof fields are switched, or perhaps the current field should not be changed until it is successfully parsed?

Note that this is a potential security issue. I presume that if the first field was an integer, an arbitrary value could be written to it which would then be interpreted as an address and passed to `free`.

Note: this was found by OSS-Fuzz on Firestore (note that I have trimmed down the repro case from the original).

PetteriAimonen commented on Mar 20, 2021 Member

Setting pData to NULL when starting to decode a pointer type oneof field sounds reasonable.
This could be done in `pb_release_union_field()`.

PetteriAimonen added Component-Decoder Priority-High Type-Defect labels on Mar 20, 2021

PetteriAimonen added a commit that referenced this issue on Mar 20, 2021

`Add testcase for #647: invalid free with oneof` 0aa1dab

PetteriAimonen added a commit that referenced this issue on Mar 20, 2021

`Fix invalid free() with oneof (#647)` ··· ✓ 4a375a5

PetteriAimonen added a commit that referenced this issue on Mar 20, 2021

`Add testcase for #647: invalid free with oneof` 9cbe4ae

PetteriAimonen added a commit that referenced this issue on Mar 20, 2021

`Fix invalid free() with oneof (#647)` ··· ✓ e2f0ccf

PetteriAimonen commented on Mar 20, 2021 Member

Security advisory: GHSA-7mv5-5mxh-qg88

I'm still a bit surprised why this hasn't been caught by the existing fuzztest. It seems the same issue should occur even with submessages, as used in `alltypes_pointer` test case, causing invalid value to be passed to `realloc()` instead of `free()` . But in any case, I should still expand the fuzz test to have string, bytes and integer fields inside the oneof.

Yet another bug that could potentially have been found with #143.

PetteriAimonen added a commit that referenced this issue on Mar 22, 2021

`fuzztest: Better coverage of static data in oneof (#647)` ··· ✕ 0a03bdf

PetteriAimonen added the FixedInGit label on Mar 22, 2021

PetteriAimonen commented on Mar 22, 2021 Member

Fix released in 0.3.9.8 and 0.4.5.

PetteriAimonen closed this as completed on Mar 22, 2021

Assignees

No one assigned

Labels

Component-Decoder FixedInGit **Priority-High** Type-Defect

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants