Read-past-end-of-buffer in libstrongswan/collections/array.c

Below is a proposed patch.

If the array is new, and you remove the last element, memmove() will read past the end of the array.

```c
static void remove_tail(array_t *array, int idx) {
    /* move all items after idx one down */
    memmove(array->data + get_size(array, idx + array->head),
    array->data + get_size(array, array->count - id[idx]);
    + get_size(array, array->count - (idx+1));
    array->count--;
    array->tail++;
}
```

Associated revisions

Revision 11f31ceb - 18.03.2014 14:46 - Tobias Brunner
array: Fix removal of elements in the second half of an array

Memory beyond the end of the array was moved when array elements in the second half of an array were removed.

Fixes #548.

History

#1 - 18.03.2014 12:10 - Tobias Brunner
- Status changed from New to Feedback
- Assignee set to Tobias Brunner

If the array is new, and you remove the last element, memmove() will read past the end of the array.

What do you mean? If it is new it is empty, so calling array_remove(array, ARRAY_TAIL, NULL) will return FALSE and does nothing else (definitely not call remove_tail()).

#2 - 18.03.2014 14:20 - Noam Lampert
This test pretty much reproduces the error:
```c
array_t* array = array_create(sizeof(int), 0);
int elem = 1;
for (int i = 0; i < 8; i++)
    array_insert(array, i, &elem);
array_remove(array, 7, NULL);
array_destroy(array);
```
at array_remove(), remove_tail() is called. This illegally copies the array to array.

#3 - 18.03.2014 14:46 - Tobias Brunner
- Tracker changed from Issue to Bug
- Status changed from Feedback to Closed
- Target version set to 5.1.3
- Resolution set to Fixed

This test pretty much reproduces the error:

Thanks for the example code. I agree, this is incorrect. Fixed with the associated commit.