**UNIVERSITY OF WATERLOO**

SE 465 - Project

Gwangseung ‘Eric’ Kim 20429117 g28kim SE465-001

Eric McAlister

Kyle Platt

March 31st, 2016

**Part (1)**  
**(b)**

**(apr\_array\_make, apr\_array\_push)**

bug: apr\_array\_push in ap\_directory\_walk, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 80.00%

bug: apr\_array\_push in ap\_add\_file\_conf, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 80.00%

bug: apr\_array\_push in ap\_add\_per\_url\_conf, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 80.00%

bug: apr\_array\_push in ap\_copy\_method\_list, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 80.00%

**bug: apr\_array\_push in apr\_xml\_insert\_uri, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 80.00%**

bug: apr\_array\_push in ap\_method\_list\_add, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 80.00%

bug: apr\_array\_push in ap\_add\_per\_dir\_conf, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 80.00%

bug: apr\_array\_push in ap\_location\_walk, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 80.00%

bug: apr\_array\_push in ap\_file\_walk, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 80.00%

bug: apr\_array\_push in set\_server\_alias, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 80.00%

bug: apr\_array\_make in ap\_init\_virtual\_host, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 86.96%

bug: apr\_array\_make in create\_core\_dir\_config, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 86.96%

bug: apr\_array\_make in apr\_xml\_parser\_create, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 86.96%

bug: apr\_array\_make in create\_core\_server\_config, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 86.96%

bug: apr\_array\_make in prep\_walk\_cache, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 86.96%

bug: apr\_array\_make in ap\_make\_method\_list, pair: (apr\_array\_make, apr\_array\_push), support: 40, confidence: 86.96%

The two functions are, as mentioned in their names, array functions that create and push to an array. The following is the source code of the highlighted ‘apr\_xml\_insert\_uri’ function in directory ‘srclib/apr-util/xml/apr\_xml.c’.

/\* return the URI's (existing) index, or insert it and return a new index \*/

APU\_DECLARE(int) apr\_xml\_insert\_uri(apr\_array\_header\_t \*uri\_array,

const char \*uri)

{

int i;

const char \*\*pelt;

/\* never insert an empty URI; this index is always APR\_XML\_NS\_NONE \*/

if (\*uri == '\0')

return APR\_XML\_NS\_NONE;

for (i = uri\_array->nelts; i--;) {

if (strcmp(uri, APR\_XML\_GET\_URI\_ITEM(uri\_array, i)) == 0)

return i;

}

pelt = apr\_array\_push(uri\_array);

\*pelt = uri; /\* assume uri is const or in a pool \*/

return uri\_array->nelts - 1;

}

The method either returns the URI’s index or insert to an already existing array passed as a parameter, so there is no need to call apr\_array\_make to create an array. The following is the source code of the make and push functions for reference, in srclib/apr/tables/apr\_tables.c

APR\_DECLARE(apr\_array\_header\_t \*) apr\_array\_make(apr\_pool\_t \*p,

int nelts, int elt\_size)

{

apr\_array\_header\_t \*res;

res = (apr\_array\_header\_t \*) apr\_palloc(p, sizeof(apr\_array\_header\_t));

make\_array\_core(res, p, nelts, elt\_size, 1);

return res;

}

...

APR\_DECLARE(void \*) apr\_array\_push(apr\_array\_header\_t \*arr)

{

if (arr->nelts == arr->nalloc) {

int new\_size = (arr->nalloc <= 0) ? 1 : arr->nalloc \* 2;

char \*new\_data;

new\_data = apr\_palloc(arr->pool, arr->elt\_size \* new\_size);

memcpy(new\_data, arr->elts, arr->nalloc \* arr->elt\_size);

memset(new\_data + arr->nalloc \* arr->elt\_size, 0,

arr->elt\_size \* (new\_size - arr->nalloc));

arr->elts = new\_data;

arr->nalloc = new\_size;

}

++arr->nelts;

return arr->elts + (arr->elt\_size \* (arr->nelts - 1));

}

**(ms\_release\_conn, strlen)**

bug: ms\_release\_conn in mc\_version\_ping, pair: (ms\_release\_conn, strlen), support: 6, confidence: 66.67%

bug: ms\_release\_conn in apr\_memcache\_stats, pair: (ms\_release\_conn, strlen), support: 6, confidence: 66.67%

**bug: ms\_release\_conn in apr\_memcache\_version, pair: (ms\_release\_conn, strlen), support: 6, confidence: 66.67%**

The following is the source code of the function apr\_memcache\_version in srclib/apr-util/memcache/apr\_memcache.c:

APU\_DECLARE(apr\_status\_t)

apr\_memcache\_version(apr\_memcache\_server\_t \*ms,

apr\_pool\_t \*p,

char \*\*baton)

{

apr\_status\_t rv;

apr\_memcache\_conn\_t \*conn;

apr\_size\_t written;

struct iovec vec[2];

rv = ms\_find\_conn(ms, &conn);

if (rv != APR\_SUCCESS) {

return rv;

}

/\* version\r\n \*/

vec[0].iov\_base = MC\_VERSION;

vec[0].iov\_len = MC\_VERSION\_LEN;

vec[1].iov\_base = MC\_EOL;

vec[1].iov\_len = MC\_EOL\_LEN;

rv = apr\_socket\_sendv(conn->sock, vec, 2, &written);

if (rv != APR\_SUCCESS) {

ms\_bad\_conn(ms, conn);

return rv;

}

rv = get\_server\_line(conn);

if (rv != APR\_SUCCESS) {

ms\_bad\_conn(ms, conn);

return rv;

}

if (strncmp(MS\_VERSION, conn->buffer, MS\_VERSION\_LEN) == 0) {

\*baton = apr\_pstrmemdup(p, conn->buffer+MS\_VERSION\_LEN+1,

conn->blen - MS\_VERSION\_LEN - 2);

rv = APR\_SUCCESS;

}

else {

rv = APR\_EGENERAL;

}

ms\_release\_conn(ms, conn);

return rv;

}

The purpose of strlen function seems to be served by MS\_VERSION\_LEN, which likely is the length of MS\_VERSION, and therefore the missing strlen function is not a bug.