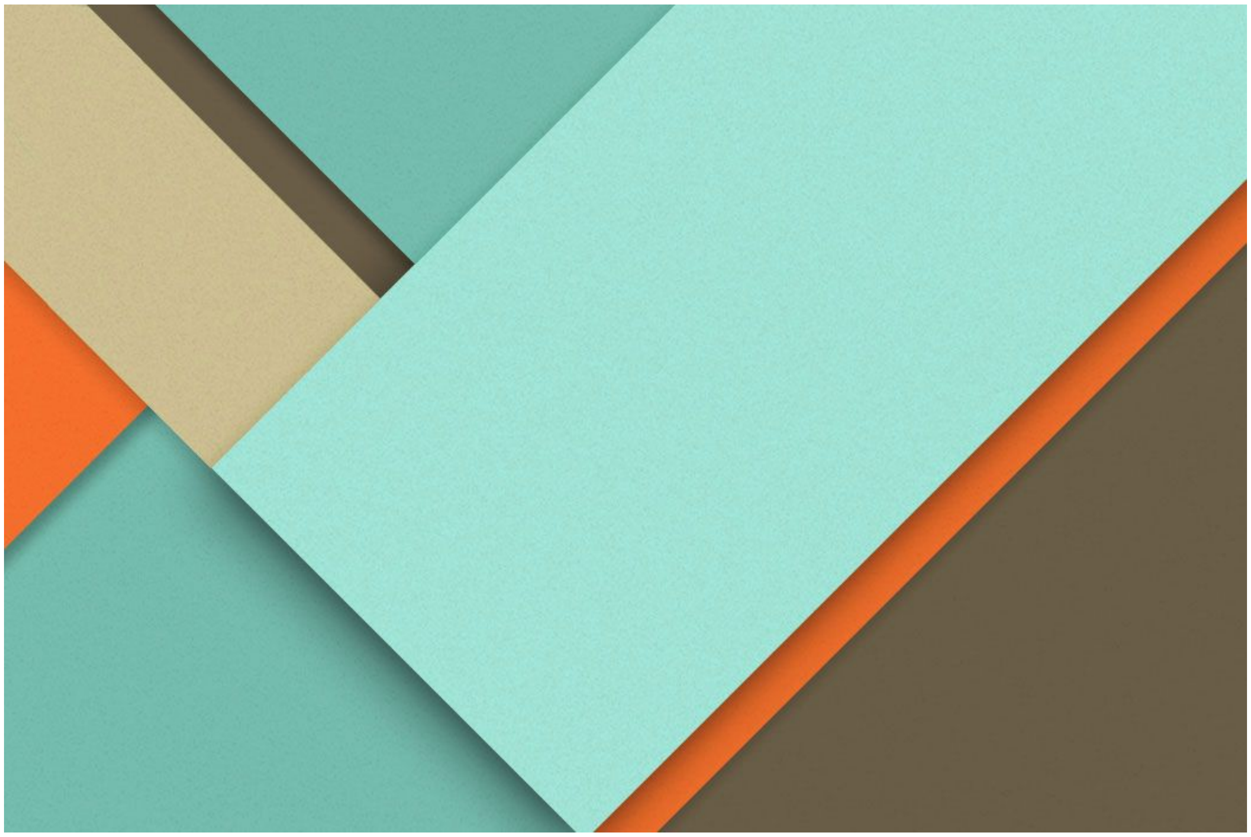


Automatic Bug-detection tool

09.04.2018

Andri Karel Júlíusson
Skúli Arnarsson
Sigurður Marteinn Lárusson

Overview

Finding and Explaining False Positives.

After assuring that all of our tests have passed for part A we used both of the given gold files to find bug in the program, we do not believe we found any real bugs but we found false positives. We then compared the found false positives to the source code of apache's http server and then tried to explain why the found errors were false positives. We looked up the source code for httpd on github and found this github page: https://github.com/apache/httpd/

Two different fundamental reasons for as to why false positives occur in general.

I. A function is called inside another scope

A false positive can occur when a pair (A, B) are expected to be called within the same function scope, however one of the elements in the pair is not called directly inside the current scope. Either function is called in a function nested within the current scope and is therefore not a bug.

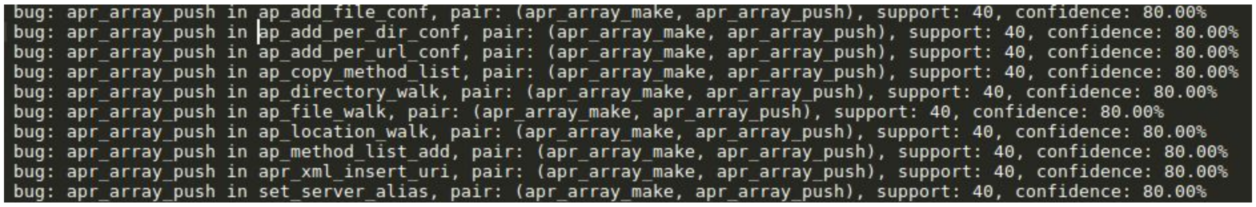
II. A pair of functions expect to be dependant but are independant

In a large program we can have to functions called often together inside the same function. A false positive may be detected when a pair of functions are expected to appear together within the same function. The bug detection tool assumes these two pairs of functions are dependent and should always be called together, if not it is registered as a false positive.

Identify 2 pairs of functions that are false positives for httpd

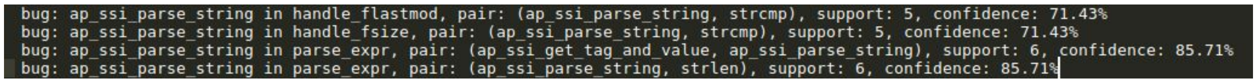
Here are the two false positives and screenshots of them from the gold files

From gold_10_80, 10 total occurrences of pair (apr_array_push, apr_array_make)



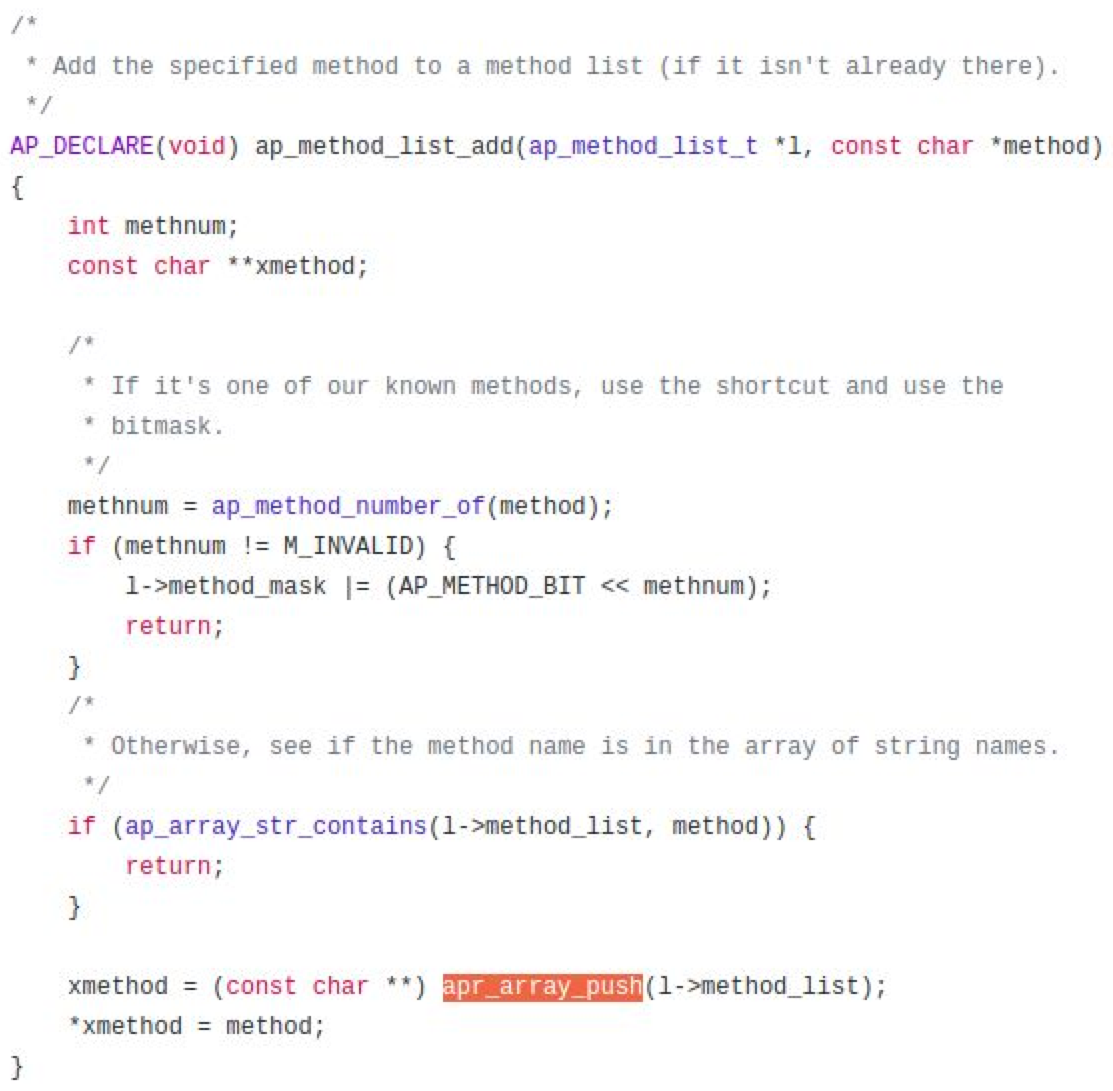
```
bug: apr_array_push in ap_add_file_conf, pair: (apr_array_make, apr_array_push), support: 40, confidence: 80.00%
bug: apr_array_push in ap_add_per_dir_conf, pair: (apr_array_make, apr_array_push), support: 40, confidence: 80.00%
bug: apr_array_push in ap_add_per_url_conf, pair: (apr_array_make, apr_array_push), 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 ap_directory_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 ap_location_walk, 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 apr_xml_insert_uri, 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%
```

From gold_3_60, 2 total occurrences of pair (ap_ssi_parse_string, strcmp)



```
bug: ap_ssi_parse_string in handle_flastmod, pair: (ap_ssi_parse_string, strcmp), support: 5, confidence: 71.43%
bug: ap_ssi_parse_string in handle_fsize, pair: (ap_ssi_parse_string, strcmp), support: 5, confidence: 71.43%
bug: ap_ssi_parse_string in parse_expr, pair: (ap_ssi_get_tag_and_value, ap_ssi_parse_string), support: 6, confidence: 85.71%
bug: ap_ssi_parse_string in parse_expr, pair: (ap_ssi_parse_string, strlen), support: 6, confidence: 85.71%
```

Why false apr_array_push produces a false positive:



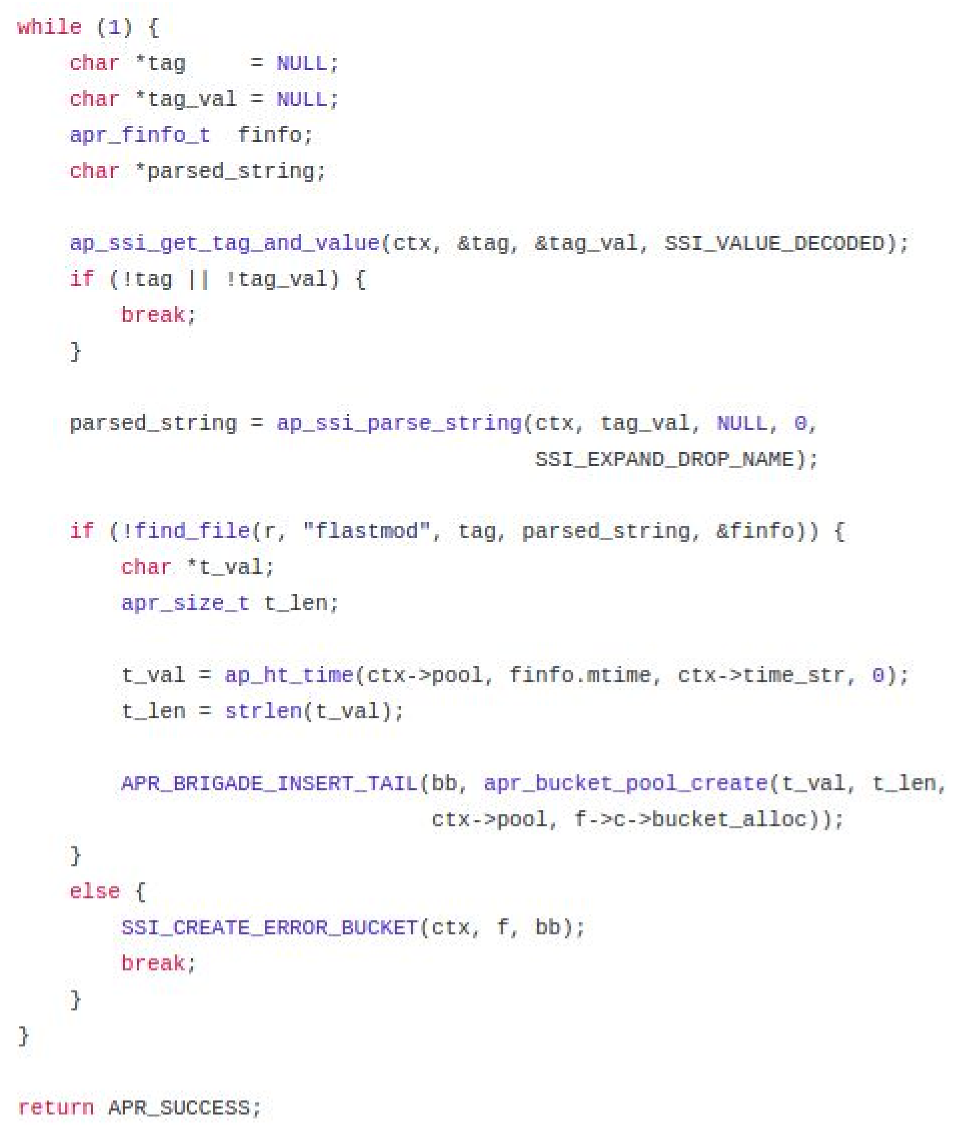
```c
/*
 * Add the specified method to a method list (if it isn't already there).
 */
AP_DECLARE(void) ap_method_list_add(ap_method_list_t *l, const char *method)
{
    int methnum;
    const char **xmethod;

    /*
     * If it's one of our known methods, use the shortcut and use the
     * bitmask.
     */
    methnum = ap_method_number_of(method);
    if (methnum != M_INVALID) {
        l->method_mask |= (AP_METHOD_BIT << methnum);
        return;
    }
    /*
     * Otherwise, see if the method name is in the array of string names.
     */
    if (ap_array_str_contains(l->method_list, method)) {
        return;
    }

    xmethod = (const char **) apr_array_push(l->method_list);
    *xmethod = method;
}
```

As we can see from this code segment. apr_array_push() is called, however apr_array_make is not present. The bug detection tool assumes that this is a bug. We can see that it is not an error because the array that is being pushed to is provided as a parameter. We check if the method is invalid before pushing to the array. Otherwise we assume that the array has been instantiated with apr_array_make before it was passed to the method as a parameter. This is a case where either function is called in a function nested within the current scope and is therefore not a bug.

Why false ap_ssi_parse_string produces a false positive:



```
while (1) {
    char *tag     = NULL;
    char *tag_val = NULL;
    apr_finfo_t  finfo;
    char *parsed_string;

    ap_ssi_get_tag_and_value(ctx, &tag, &tag_val, SSI_VALUE_DECODED);
    if (!tag || !tag_val) {
        break;
    }

    parsed_string = ap_ssi_parse_string(ctx, tag_val, NULL, 0,
                                        SSI_EXPAND_DROP_NAME);

    if (!find_file(r, "flastmod", tag, parsed_string, &finfo)) {
        char *t_val;
        apr_size_t t_len;

        t_val = ap_ht_time(ctx->pool, finfo.mtime, ctx->time_str, 0);
        t_len = strlen(t_val);

        APR_BRIGADE_INSERT_TAIL(bb, apr_bucket_pool_create(t_val, t_len,
                                ctx->pool, f->c->bucket_alloc));
    }
    else {
        SSI_CREATE_ERROR_BUCKET(ctx, f, bb);
        break;
    }
}

return APR_SUCCESS;
```

If we look at the function handle_flastmod, where the pair (ap_ssi_parse_string, strcmp) is expected to appear together. However in this case it does not. As we well know strcmp, that is used for string comparisons in C is in no way dependent on ap_ssi_parse_string. The same goes for ap_ssi_parse_string where in this example when parsing a ssi string, strcmp is not needed in order to parse strings. In on multiple occasions strcmp is used in the same functions as ap_ssi_parse_string and oftentimes these functions are extremely long and handle parsing strings and therefore it is not surprising that this pair of functions appear together more often than not, although not being dependant on each other.

This example is a case of when a pair of functions expect to be dependant but are independant.