

PVS-Studio is ready to improve the code of Tizen operating system



Andrey Karpov.

CTO

karpov@viva64.com

2017

Tizen



- Open operating system based of the Linux kernel.
- Samsung company cares about the quality, reliability and safety of the code. In particular:
 - Samsung company uses Svace static code analyzer, developed by Institute for System Programming of the Russian Academy of Sciences (ISP RAS).
 - This is the first and the only mobile operational system in Russia, certified in the Federal Service for Technical and Export Control (FSTEC) – noted a representative from Samsung.



Objective

- Contract agreement with PVS-Studio team concerning the error fixing and regular code audit.
- Which errors is PVS-Studio able to find?
- How many errors can PVS-Studio team fix?

The preliminary study was completed by:

- candidate of physicomathematical sciences
Andrey Nikolaevich Karpov, 1981;
- Technical director of "Program Verification Systems" CoLtd.,
- MVP in the Visual C++ category 
- Intel Black Belt Software Developer  Intel® Black Belt Software Developer
- One of the founders of the PVS-Studio project
(static code analyzer for C/C++/C#).
- Participated in the analysis of more than a
hundred of open source projects



Important

- The presentation shows the amount of errors that can be fixed after the analysis of Tizen using PVS-Studio.
- However, while a one-time bug fix is a necessary action, it is not quite sufficient for the overall code improvement.
- In a one-time analysis, a reviewer usually finds the errors which do not seriously affect the functionality of a program. Therefore, any single check can demonstrate the abilities of the analyzer, but only so much.
- The quality and reliability of the project increases only with the regular use of static code analyzer.
- The earlier an error is detected, the less expensive it is to correct it.

I believe that:

- Currently, PVS-Studio detects more than 10% of errors that are present in the code of the Tizen project.
- In the case of regular use of PVS-Studio on the new code, about 20% of errors can be prevented.
- I predict that PVS-Studio team can detect and fix about 27 000 errors in the Tizen project.
- The reasoning of the evaluations will be provided further on.

The conditions of the study

- I worked with the source code
<https://build.tizen.org/project/show/Tizen:Unified>.
- To assess the abilities of PVS-Studio analyzer, I examined the analyzer reports of various projects and noted down the bugs that were found.
- The projects were selected randomly.
- I've divided the projects into two groups:
 - The projects developed by the Samsung specialists
 - Third-party projects

Let's consider the error types that seemed most important and interesting to me

- Once again, let me emphasize that this is not about the number of warnings issued by the analyzer, but about real errors.

Projects developed by Samsung specialists

- bluetooth-frwk-0.2.157
- capi-appfw-application-0.5.5
- capi-base-utils-3.0.0
- capi-content-media-content-0.3.10
- capi-maps-service-0.6.12
- capi-media-audio-io-0.3.70
- capi-media-codec-0.5.3
- capi-media-image-util-0.1.15
- capi-media-player-0.3.58
- capi-media-screen-mirroring-0.1.78
- capi-media-streamrecorder-0.0.10
- capi-media-vision-0.3.24
- capi-network-bluetooth-0.3.4
- capi-network-http-0.0.23
- cynara-0.14.10
- e-mod-tizen-devicemgr-0.1.69
- ise-engine-default-1.0.7
- ise-engine-sunpinyin-1.0.10
- ise-engine-tables-1.0.10

V501. A typo: the variable is compared with itself

```
bool operator <(const TSegment& other) const {  
    if (m_start < other.m_start)  
        return true;  
  
    if (m_start == other.m_start)  
        return m_len < m_len;  
  
    return false;  
}
```

- V501 There are identical sub-expressions to the left and to the right of the '<' operator: m_len < m_len segmentor.h 65
- It should be: m_len < other.m_len
- Errors in total: 2

V503. A typo: meaningless comparison

```
int *focus_unit = (int *)data;
if (focus_unit == NULL || focus_unit < 0) {
    _E("focus page is wrong");
    return ;
}
```

- V503 This is a nonsensical comparison: pointer < 0.
apps_view_circle_indicator.c 193
- Should be: *focus_unit < 0
- Errors in total: 2

V507. A non-existing buffer being used

```
void extract_input_aacdec_m4a_test(  
    ..., unsigned char **data, ...)   
{  
    unsigned char buffer[100000];  
    ....  
DONE:  
    *data = buffer;  
    *have_frame = TRUE;  
    if (read_size >= offset) *size = offset;  
    else *size = read_size;  
}
```

- V507 Pointer to local array 'buffer' is stored outside the scope of this array. Such a pointer will become invalid. media_codec_test.c 793
- Errors in total: 1

V512. Incorrect work with the buffer: more elements are processed than it is necessary

```
typedef int gint;  
typedef gint gboolean;
```

```
#define BT_REQUEST_ID_RANGE_MAX 245
```

```
static gboolean req_id_used[BT_REQUEST_ID_RANGE_MAX];
```

```
memset(req_id_used, 0x00, BT_REQUEST_ID_RANGE_MAX);
```

- V512 A call of the 'memset' function will lead to underflow of the buffer 'req_id_used'. bt-service-util.c
- Should be: sizeof(req_id_used)

V512. Incorrect work with the buffer: potential buffer index out of bounds

```
char buf[256] = "\0";
....
snprintf(buf, sizeof(buf), "%s, %s, ",
        name, _("IDS_BR_BODY_IMAGE_T_TTS"));
....
snprintf(buf + strlen(buf), sizeof(buf),
        "%s, ", _("IDS_ACCS_BODY_SELECTED_TTS"));
```

- V512 A call of the 'memset' function will lead to underflow of the buffer 'req_id_used'. bt-service-util.c
- Should be: `sizeof(buf) - strlen(buf)`

V512. Incorrect work with the buffer: potential buffer index out of bounds

```
#define BT_ADDRESS_STRING_SIZE 18

typedef struct { unsigned char addr[6]; } bluetooth_device_address_t;

typedef struct {
    int count;
    bluetooth_device_address_t addresses[20];
} bt_dpm_device_list_t;

bt_dpm_device_list_t device_list;
for (; list; list = list->next, i++) {
    memset(device_list.addresses[i].addr, 0, BT_ADDRESS_STRING_SIZE);
    ....
}
```

V512. Incorrect work with the buffer

- The error, described in the previous slide, is detected thanks to the warning: V512 A call of the 'memset' function will lead to overflow of the buffer 'device_list.addresses[i].addr'. bt-service-dpm.c 226
- Errors in total: 7

V517. A logic error in the sequences if .. else .. if

```
#define LANG_ES_MX "\x45\x73\x70\x61\xC3\xB1\x6f\x6c\x20\x28\" \
  "\x45\x73\x74\x61\x64\x6f\x73\x20\x55\x6e\x69\x64\x6f\x73\x29"

#define LANG_ES_US "\x45\x73\x70\x61\xC3\xB1\x6f\x6c\x20\x28\" \
  "\x45\x73\x74\x61\x64\x6f\x73\x20\x55\x6e\x69\x64\x6f\x73\x29"
```

Similar strings

```
} else if (!strcmp(LANG_PT_PT, lang)) {return "pt_PT"; }
  else if (!strcmp(LANG_ES_MX, lang)) { return "es_MX"; }
  else if (!strcmp(LANG_ES_US, lang)) { return "es_US"; }
  else if (!strcmp(LANG_EL_GR, lang)) { return "el_GR"; }
```

- V517 The use of 'if (A) {...} else if (A) {...}' pattern was detected. There is a probability of logical error presence. Check lines: 144, 146.
voice_setting_language.c 144
- Errors in total: 4

V519. Repeated assignment (error in the logic of the program)

```
WSCContextISF* old_focused = _focused_ic;  
_focused_ic = context_scim;  
_focused_ic = old_focused;
```

- V519 The '_focused_ic' variable is assigned values twice successively. Perhaps this is a mistake. Check lines: 1260, 1261.
wayland_panel_agent_module.cpp 1261
- Perhaps, this is an incorrect “swap” and it should be:

```
WSCContextISF* old_focused = _focused_ic;  
_focused_ic = context_scim;  
context_scim = old_focused;
```

V519. Repeated assignment (a typo)

```
Elm_Genlist_Item_Class *ttc = elm_genlist_item_class_new();  
Elm_Genlist_Item_Class *mtc = elm_genlist_item_class_new();
```

```
ttc->item_style = "title";  
ttc->func.text_get = gl_title_text_get_cb;  
ttc->func.del = gl_del_cb;
```

```
mtc->item_style = "multiline";  
mtc->func.text_get = gl_multi_text_get_cb;  
ttc->func.del = gl_del_cb;
```

- V519 The 'ttc->func.del' variable is assigned values twice successively. Perhaps this is a mistake. Check lines: 409, 416.
privacy_package_list_view.c 416

V519. Repeated assignments (unaccounted)

- I noticed 11 errors.
- However, I ignored a large number of cases when a status is overwritten in the code like this:

```
status = Foo(1);  
status = Foo(2);  
status = Foo(3);
```
- If we take such cases into account, there will be much more errors.

V522. There is no check of a pointer

- A pointer is dereferenced without a preliminary check.
- At the same time, a pointer can be null, as it is obtained in one of the following ways:
 - `p = (type)malloc(n);`
 - `p = strdup(s);`
 - `p = dynamic_cast<type>(q);`
 - `p = strstr(s, "qwerty");`
 - and so on.
- Note. In many other places in the code, the memory after `malloc/realloc/...` is checked. Therefore, it is highly likely that in these detected fragments the check is forgotten and the code should be fixed.
- Errors in total: 73

V522. The pointer can be null (malloc)

```
Edje_Message_Int_Set* msg =  
    (Edje_Message_Int_Set *)malloc(sizeof(*msg) + 3 * sizeof(int));  
msg->count = 4;  
msg->val[0] = r;  
msg->val[1] = g;  
msg->val[2] = b;  
msg->val[3] = a;
```

- V522 There might be dereferencing of a potential null pointer 'msg'.
QuickAccess.cpp 743

V522. The pointer can be null (dynamic_cast)

```
CAudioInput* inputHandle =  
    dynamic_cast<CAudioInput*>(handle->audioIoHandle);  
assert(inputHandle);  
inputHandle->peek(buffer, &_length);
```

- V522 There might be dereferencing of a potential null pointer 'inputHandle'. cpp_audio_io.cpp 928
- A strange code. If we are sure that this is CAudioInput, then we should use static_cast. And if you are not sure, we need a check. The assert macro won't help in the release version.

V575. Similarly. A pointer can be null upon the call of the strncpy function

```
char *temp1 = strstr(dp->d_name, "-");  
char *temp2 = strstr(dp->d_name, ".");
```

```
strncpy(temp_filename, dp->d_name, strlen(dp->d_name)-strlen(temp1));  
strncpy(file_format, temp2, strlen(temp2));
```

- V575 The potential null pointer is passed into 'strlen' function. Inspect the first argument. image_util_decode_encode_testsuite.c 207
- V575 The potential null pointer is passed into 'strlen' function. Inspect the first argument. image_util_decode_encode_testsuite.c 208

V575. The pointer can be null upon the call of the memcpy function

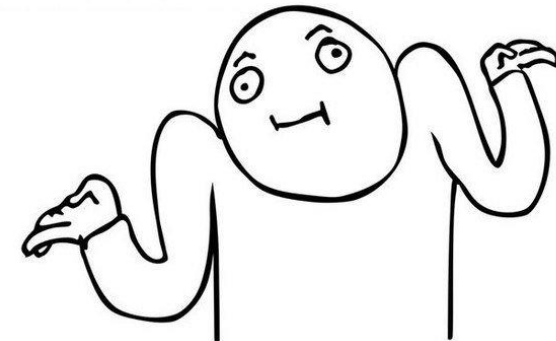
```
uint32_t tlen = strlen (text), ilen = strlen (insert);  
char *new_text = (char*)malloc (tlen + ilen + 1);  
if ((unsigned int) tlen < offset)  
    offset = tlen;  
memcpy (new_text, text, offset);
```

- V575 The potential null pointer is passed into 'memcpy' function. Inspect the first argument. wayland_panel_agent_module.cpp 1060
- Errors in total: 15

Note. There is a check where it is not much needed. This and the previous slides refer to the same project.

```
static FilterModule *__filter_modules    = 0;

static void
__initialize_modules (const ConfigPointer &config)
{
    ....
    __filter_modules = new FilterModule [__number_of_modules];
    if (!__filter_modules) return;
    ....
}
```



V523. The action doesn't depend on the condition.

```
static void _content_resize(..., const char *signal)
{
    ....
    if (strcmp(signal, "portrait") == 0) {
        evas_object_size_hint_min_set(s_info.layout,
            ELM_SCALE_SIZE(width), ELM_SCALE_SIZE(height));
    } else {
        evas_object_size_hint_min_set(s_info.layout,
            ELM_SCALE_SIZE(width), ELM_SCALE_SIZE(height));
    }
    ....
}
```



Identical actions

- V523 The 'then' statement is equivalent to the 'else' statement.
page_setting_all.c 125

V527. The pointer was not dereferenced

```
int _read_request_body(http_transaction_h http_transaction,
                      char **body)
{
    ....
    memcpy(*body + curr_len, ptr, body_size);
    body[new_len] = '\0';
    curr_len = new_len;
    ....
}
```

- V527 It is odd that the '\0' value is assigned to 'char' type pointer. Probably meant: *body[new_len] = '\0'. http_request.c 370
- Correct code: (*body)[new_len] = '\0';
- Errors in total: 1

V547. The condition is always true/false

```
unsigned m_candiPageFirst;
```

```
bool
```

```
CIMIClassicView::onKeyEvent(const CKeyEvent& key)
```

```
{
```

```
....
```

```
if (m_candiPageFirst > 0) {
```

```
    m_candiPageFirst -= m_candiWindowSize;
```

```
    if (m_candiPageFirst < 0) m_candiPageFirst = 0;
```

```
    changeMasks |= CANDIDATE_MASK;
```

```
}
```

- V547 Expression 'm_candiPageFirst < 0' is always false. Unsigned type value is never < 0. imi_view_classic.cpp 201
- Errors in total: 9

V560. A part of the condition is always true/false

```
unsigned char val, zero_count, i;
....
val = buffer[0];
while (!val) {
    if ((zero_count == 2 || zero_count == 3) && val == 1)
        break;
    ....
}
```

- V560 A part of conditional expression is always false: val == 1.
player_es_push_test.c 284
- Errors in total: 2

V572. Confusion between types of created and destroyed objects

- There are three structs that aren't related to each other at all:

```
struct sockaddr_un
{
    sa_family_t sun_family;
    char sun_path[108];
};
```

```
struct sockaddr
{
    sa_family_t sa_family;
    char sa_data[14];
};
```

```
struct sockaddr_in {
    sa_family_t sin_family;
    in_port_t sin_port;
    struct in_addr sin_addr;
    unsigned char sin_zero[sizeof (struct sockaddr) -
        (sizeof (unsigned short int)) - sizeof (in_port_t) -
        sizeof (struct in_addr)];
};
```

```

class SocketAddress::SocketAddressImpl
{
    struct sockaddr *m_data;
    ....
    SocketAddressImpl (const SocketAddressImpl &other)
    {
        ....
        case SCIM_SOCKET_LOCAL:
            m_data = (struct sockaddr*) new struct sockaddr_un;
            len = sizeof (sockaddr_un);
            break;
        case SCIM_SOCKET_INET:
            m_data = (struct sockaddr*) new struct sockaddr_in;
            len = sizeof (sockaddr_in);
            break;
        ....
    }

    ~SocketAddressImpl () {
        if (m_data) delete m_data;
    }
};

```


- Warnings:
 - V572 It is odd that the object which was created using 'new' operator is immediately cast to another type. scim_socket.cpp 136
 - V572 It is odd that the object which was created using 'new' operator is immediately cast to another type. scim_socket.cpp 140
- Errors in total: 4

V595. The pointer is checked only after it was already dereferenced

```
static void _show(void *data)
{
    SETTING_TRACE_BEGIN;
    struct _priv *priv = (struct _priv *)data;
    Eina_List *children = elm_box_children_get(priv->box);
    Evas_Object *first = eina_list_data_get(children);
    Evas_Object *selected = eina_list_nth(children,
                                           priv->item_selected_on_show);

    if (!priv) {
        _ERR("Invalid parameter.");
        return;
    }
}
```

- V595 The 'priv' pointer was utilized before it was verified against nullptr. Check lines: 110, 114. view_generic_popup.c 110
- Errors in total: 5

V597. Private data is not cleared

```
static void SHA1Final(unsigned char digest[20], SHA1_CTX* context)
{
    u32 i;
    unsigned char finalcount[8];
    ....
    memset(context->count, 0, 8);
    memset(finalcount, 0, 8);
}
```

- V597 The compiler could delete the 'memset' function call, which is used to flush 'finalcount' buffer. The memset_s() function should be used to erase the private data. wifi_generate_pin.c 185
- Errors in total: 1

V611. Confusion with the allocation and freeing of the memory

```
char *full_path = NULL;
....
full_path = (char *)alloca(PATH_MAX);
....
if (!select_all_item) {
    SETTING_TRACE_ERROR("select_all_item is NULL");
    free(full_path);
    return;
}
```

- V611 The memory was allocated using 'alloca' function but was released using the 'free' function. Consider inspecting operation logics behind the 'full_path' variable. setting-ringtone-remove.c 88
- Errors in total: 2

V614. A potentially uninitialized variable

```
Eext_Circle_Surface *surface;  
....  
if (_WEARABLE)  
    surface = eext_circle_surface_conformant_add(conform);  
....  
app_data->circle_surface = surface;
```

- V614 Potentially uninitialized pointer 'surface' used.
w-input-selector.cpp 896
- Errors in total: 1

V636. Incorrect operations of division

```
static void
_e_input_devmgr_request_client_add(..., uint32_t duration)
{
    struct wl_listener *destroy_listener = NULL;
    double milli_duration = duration / 1000;
    ....
}
```

- V636 The 'duration / 1000' expression was implicitly cast from 'int' type to 'double' type. Consider utilizing an explicit type cast to avoid the loss of a fractional part. An example: double A = (double)(X) / Y;. e_devicemgr_device.c 648
- Apparently, it should be: (double)(duration) / 1000
- Errors in total: 4

V640. The code's operational logic does not correspond with its formatting

- The reason is a poorly written macro

```
#define MC_FREEIF(x) \  
    if (x) \  
        g_free(x); \  
x = NULL;
```



- Here is the way the macro is used:

```
static gboolean __mc_gst_init_gstreamer()
{
    int i = 0;
    ....
    for (i = 0; i < arg_count; i++)
        MC_FREEIF(argv2[i]);
    ....
}
```

- V640 The code's operational logic does not correspond with its formatting. The second statement will always be executed. It is possible that curly brackets are missing. media_codec_port_gst.c
1800

- After expanding the macro we get:

```
for (i = 0; i < arg_count; i++)  
    if (argv2[i])  
        g_free(argv2[i]);  
argv2[i] = NULL;
```

- As a result:
 - the pointers won't be nullified
 - NULL will be written outside the array bound
- Errors in total: 2

V642. Loss of significant bits

```
typedef unsigned char Eina_Bool;

static Eina_Bool _state_get(...)
{
    ....
    if (!strcmp(part, STATE_BROWSER))
        return !strcmp(id, APP_ID_BROWSER);
    else if (!strcmp(part, STATE_NOT_BROWSER))
        return strcmp(id, APP_ID_BROWSER);
    ....
}
```

- V642 Saving the 'strcmp' function result inside the 'unsigned char' type variable is inappropriate. The significant bits could be lost breaking the program's logic. grid.c 137

```
typedef unsigned char Eina_Bool;
static Eina_Bool _state_get(...)
{
    ....
    if (!strcmp(part, STATE_BROWSER))
        return !strcmp(id, APP_ID_BROWSER);
    else if (!strcmp(part, STATE_NOT_BROWSER))
        return strcmp(id, APP_ID_BROWSER);
}
```

- Isn't a complementary operator forgotten here? We see it in other fragments.
- Even if it is not forgotten, the code is still quite bad.
- The result of the 'int' type is cut to 'unsigned char'. This code can be a source of a vulnerability (see the description of the diagnostic V642).
- Errors in total: 1

V645. Off-by-one Error

```
#define OP_MAX_URI_LEN 2048
char object_uri[OP_MAX_URI_LEN];

strncat(dd_info->object_uri, ch_str,
        OP_MAX_URI_LEN - strlen(dd_info->object_uri));
```

- V645 The 'strncat' function call could lead to the 'dd_info->object_uri' buffer overflow. The bounds should not contain the size of the buffer, but a number of characters it can hold. oma-parser-dd1.c 422
- 1 should be subtracted
- Errors in total: 2

V647. Treacherous C language. An undeclared function is used

```
int *labels = malloc(sizeof(int) * number_of_persons);
```

- The error is detected indirectly.
- V647 The value of 'int' type is assigned to the pointer of 'int' type.
surveillance_test_suite.c 928
- The malloc function is not declared anywhere (the header file is not included).
- If Tizen becomes 64-bit, it will be a problem. Higher bits of the pointer will be lost, as it is presupposed by default that the function returns the 'int' type.
- Errors in total: 1

V668. It is not taken into account that the 'new' operator, as opposed to malloc, does not return NULL (not a dangerous case)

```
template <class T> class vector {  
private:  
    ....  
    void push_back(const T &value)  
    {  
        T *clone = new T(value);  
        if (clone) {  
            g_array_append_val(parray, clone);  
            current_size++;  
        }  
    }  
};
```

- V668 There is no sense in testing the 'clone' pointer against null, as the memory was allocated using the 'new' operator. The exception will be generated in the case of memory allocation error. maps_util.h

153

V668. It is not taken into account that the 'new' operator, as opposed to malloc, does not return NULL (a dangerous case)

```
bool CThreadSlm::load(const char* fname, bool MMap)
{
    int fd = open(fname, O_RDONLY);
    ....
    if ((m_buf = new char[m_bufSize]) == NULL) {
        close(fd);
        return false;
    }
    ....
}
```

- V668 There is no sense in testing the 'm_buf' pointer against null, as the memory was allocated using the 'new' operator. The exception will be generated in the case of memory allocation error. slm.cpp 97
- Errors in total: 54

V674. Confusion between integer and real

```
static void preview_down_cb(...)\n{\n    ....\n    int delay = 0.5;\n    double fdelay;\n    fdelay = ((double)delay / 1000.0f);\n    DbgPrint("Long press: %lf\\n", fdelay);\n\n    //delay = SYSTEM_SETTINGS_TAP_AND_HOLD_DELAY_SHORT; /* default 0.5 sec */\n    //if (system_settings_get_value_int(SYSTEM_SETTINGS_KEY_TAP_AND_HOLD_DELAY, &delay) != 0) {\n    //delay = SYSTEM_SETTINGS_TAP_AND_HOLD_DELAY_SHORT;\n    //}\n\n    cbdata->long_press_timer = ecore_timer_add(fdelay, long_press_cb, cbdata);
```

- V674 The '0.5' literal of the 'double' type is assigned to a variable of the 'int' type. Consider inspecting the '= 0.5' expression. add-viewer.c 824

- Most likely we are dealing with unsuccessful refactoring.
- A programmer decided to comment a part of the code and make the fdelay variable always equal 0.5.
- I.e. the code was probably meant to be like this:

```
static void preview_down_cb(....)
{
    ....
    int delay = 500;
    double fdelay;
    fdelay = ((double)delay / 1000.0f);
    DbgPrint("Long press: %lf\n", fdelay);
}
```

- Errors in total: 1

V675. Writing to the read-only memory (luckily, this code is taken from the tests)

```
int test_batch_operations()
{
    ....
    char *condition = "MEDIA_PATH LIKE \';
    strncat(condition, tzplatform_mkpath(TZ_USER_CONTENT,
                                          "test/image%%jpg\'"), 17);
    ....
}
```

- V675 Calling the 'strncat' function will cause the writing into the read-only memory. Inspect the first argument. media-content_test.c 2952
- Errors in total: 1

V696. Incorrect loops

```
do {  
    ret = TEMP_FAILURE_RETRY(getpwnam_r(...));  
    if (ret == ERANGE && buffer.size() < MEMORY_LIMIT) {  
        buffer.resize(buffer.size() << 1);  
        continue;  
    }  
} while (0);
```

- V668 There is no sense in testing the 'm_buf' pointer against null, as the memory was allocated using the 'new' operator. The exception will be generated in the case of memory allocation error. slm.cpp 97
- The 'continue' operator will exit the loop, not resume it.
- Errors in total: 2

V701. A dangerous way to use realloc (it can lead to a memory leak)

```
value = (char *) realloc(value, value_size);  
if (value == NULL) {  
    func_ret = PREFERENCE_ERROR_OUT_OF_MEMORY;  
    break;  
}
```

- V701 realloc() possible leak: when realloc() fails in allocating memory, original pointer 'value' is lost. Consider assigning realloc() to a temporary pointer. preference.c 951
- The old value of the pointer is not retained. If realloc doesn't reallocate the memory, then a memory leak will occur.
- Errors in total: 11

V773. Memory leak

- First, let's consider three functions that are used. It is important for us that they all will return a pointer to the allocated memory.

```
char *generate_role_trait(AtspiAccessible * obj) {  
    ....  
    return strdup(ret);  
}
```

```
char *generate_description_trait(AtspiAccessible * obj) {  
    ....  
    return strdup(ret);  
}
```

```
char *generate_state_trait(AtspiAccessible * obj) {  
    ....  
    return strdup(ret);  
}
```

```
char *role_name = generate_role_trait(obj);  
char *description_from_role = generate_description_trait(obj);  
char *state_from_role = generate_state_trait(obj);  
....  
char *desc = atspi_accessible_get_description(obj, &err);
```

```
if (err)  
{  
    g_error_free(err);  
    g_free(desc);  
    return strdup(trait);  
}
```

- V773 The function was exited without releasing the 'role_name' pointer. A memory leak is possible. navigator.c 991
- V773 The function was exited without releasing the 'description_from_role' pointer. A memory leak is possible. navigator.c 991
- V773 The function was exited without releasing the 'state_from_role' pointer. A memory leak is possible. navigator.c 991

- In case of an error, not all of the memory is freed
- Errors in total: 3

V778. A typo in the identical code blocks

```
if (m_modulesToolbar) {  
    evas_object_smart_callback_del(m_modulesToolbar,  
        "language,changed", _modules_toolbar_language_changed);  
    evas_object_del(m_modulesToolbar);  
}  
if (m_navigatorToolbar) {  
    evas_object_smart_callback_del(m_navigatorToolbar,  
        "language,changed", _navigation_toolbar_language_changed);  
    evas_object_del(m_modulesToolbar);  
}
```

- V778 Two similar code fragments were found. Perhaps, this is a typo and 'm_navigatorToolbar' variable should be used instead of 'm_modulesToolbar'. BookmarkManagerUI.cpp 66
- Errors in total: 1

V779. Dead code

```
static bool __check_myplace_automation(void)
{
    LS_FUNC_ENTER
    bool myplace_automation_supported = false;
    bool myplace_automation_consent = false;
    ....
    return false;
    LS_FUNC_EXIT
}
```

- V779 Unreachable code detected. It is possible that an error is present. myplace-suggest.c 68
- Errors in total: 8

V780. Incorrect initialization of objects

```
struct _VoiceData {  
    ....  
    std::vector<std::string> stt_results;  
    ....  
};  
typedef struct _VoiceData VoiceData;  
  
my_voicedata = (VoiceData*)malloc(sizeof(VoiceData));  
....  
memset(my_voicedata, 0, sizeof(VoiceData));
```

- V780 The object 'my_voicedata' of a non-passive (non-PDS) type cannot be initialized using the memset function. ise-stt-mode.cpp 773
- Errors in total: 2

Other errors:

- V505. Errors in total: 1
- V523. Errors in total: 6
- V524. Errors in total: 1
- V535. Errors in total: 4
- V556. Errors in total: 18
- V571. Errors in total: 1
- V576. Errors in total: 4
- V618. Errors in total: 6
- V622. Errors in total: 1
- V624. Errors in total: 2
- V646. Errors in total: 2
- V686. Errors in total: 1
- V690. Errors in total: 7

Expected density of errors in the code of Tizen

- According to the information of the researchers from Carnegie-Mellon university, 1000 lines of code contain 5-15 errors.
- It is generally thought that Linux and its components have less than 1 error per 1000 lines of code.
- Tizen developers also care about the quality of the code.
- Let's choose a pessimistic approach.
- I suppose that in the Tizen code there are 3 errors per 1000 lines of code.
- Yes, I can be very wrong, but we need to calculate it somehow.

The percentage of errors, detected by PVS-Studio

- We performed the analysis of 1 036 000 lines of code
- The percentage of comments is 19.9%
- I detected **345** errors.
- So it turns out that PVS-Studio detects 0.41 errors per 1000 lines of code.
- If Tizen has 3 errors per 1000 lines of code, then PVS-Studio analyzer can detect more than 10% of undetected errors.
- This percent will be higher for the new code that will be written further on. We can safely say that PVS-Studio analyzer can prevent about 20% of errors.

Now let's speak about the analysis of third-party libraries

- alsa-lib-1.0.28
- aspell-0.60.6.1
- augeas-1.3.0
- bind-9.11.0
- efl-1.16.0
- enlightenment-0.20.0
- ise-engine-anthy-1.0.9

- The code of third-party libraries is equally important.
- A phone user doesn't care if the vulnerability or the memory leak was in the third-party library or not.
- To cut the story short, here are several most interesting errors, in my opinion.

V501. A typo in a complex condition (quite often people don't think about such errors, but they exist)

```
for (i = 0; i < pd->map.colors_count; ++i)
{
    if ((pd->map.colors[i]->r != 255) ||
        (pd->map.colors[i]->g != 255) ||
        (pd->map.colors[i]->b != 255) ||
        (pd->map.colors[i]->b != 255))
        . . . .
```

- V501 There are identical sub-expressions '(pd->map.colors[i]->b != 255)' to the left and to the right of the '||' operator. edge_edit.c 14052
- A blue component was rechecked instead of the alpha channel.
- Errors in total: 5

V522. This diagnostic detects not only potential, but also the explicit null pointer dereference

```
static isc_result_t setup_style(dns_master_style_t **stylep) {  
    ....  
    REQUIRE (stylep != NULL || *stylep == NULL);  
}
```

- V522 Dereferencing of the null pointer 'stylep' might take place. Check the logical condition. delv.c 500
- Perhaps it should be: (stylep != NULL && *stylep == NULL);
- Errors in total: 203

V591. The function returns a random value

```
static Eina_Bool _ipc_server_data(...)  
{  
    ....  
    //TIZEN_ONLY(170317): add skipping indicator buffer logic  
    if (indicator_buffer_skip)  
        return;  
    //END  
    ....
```



- V591 Non-void function should return a value. ecore_evas_extn.c 1526
- An example of a bad patch for a third-party library
- Errors in total: 5

V774. Using the freed memory

```
if (ctx != NULL) {  
    char *c = realloc(child, strlen(child)-strlen(ctx)+1);  
    if (c == NULL)  
        return NULL;  
    int ctxidx = strlen(ctx);  
    if (child[ctxidx] == SEP)  
        ctxidx++;  
    strcpy(c, &child[ctxidx]);  
    child = c;  
}
```

- V774 The 'child' pointer was used after the memory was reallocated.
augtool.c 151
- Errors in total: 7

V778. PVS-Studio is really good at finding bad code, caused by sloppy Copy-Paste

```
void Config::del()
{
    while (first_) {
        Entry * tmp = first_>next;
        delete first_;
        first_ = tmp;
    }
    while (others_) {
        Entry * tmp = others_>next;
        delete first_;
        others_ = tmp;
    }
    ....
}
```

- V778 Two similar code fragments were found. Perhaps, this is a typo and 'others_' variable should be used instead of 'first_'. config.cpp 185
- Errors in total: 2

Other errors in the third-party libraries

- V502. Errors in total: 1
- V505. Errors in total: 25
- V517. Errors in total: 4
- V519. Errors in total: 3
- V523. Errors in total: 2
- V528. Errors in total: 1
- V541. Errors in total: 1
- V547. Errors in total: 10
- V556. Errors in total: 6
- V571. Errors in total: 1
- V575. Errors in total: 67
- V576. Errors in total: 1
- V590. Errors in total: 3

Analysis results of the third party libraries

- 1 915 000 lines of code were analyzed
- Among them, comments are 17,6%
- I detected **564** errors.
- It turns out that PVS-Studio detects 0.36 errors per 1000 lines of code.
- Why is the error density lower in the libraries?
 - I may have studied the code less attentively and haven't noticed all the errors.
 - A lot of projects are already regularly checked by Coverity.

Overall Results

V501	There are identical sub-expressions to the left and to the right of the 'foo' operator.	6
V502	Perhaps the '?:' operator works in a different way than it was expected. The '?:' operator has a lower priority than the 'foo' operator.	1
V503	This is a nonsensical comparison: pointer < 0.	2
V505	The 'alloca' function is used inside the loop. This can quickly overflow stack.	26
V507	Pointer to local array 'X' is stored outside the scope of this array. Such a pointer will become invalid.	1
V512	A call of the 'Foo' function will lead to a buffer overflow or underflow.	7
V517	The use of 'if (A) {...} else if (A) {...}' pattern was detected. There is a probability of logical error presence.	8
V519	The 'x' variable is assigned values twice successively. Perhaps this is a mistake.	14
V522	Dereferencing of the null pointer might take place.	276
V523	The 'then' statement is equivalent to the 'else' statement.	8
V524	It is odd that the body of 'Foo_1' function is fully equivalent to the body of 'Foo_2' function.	1
V527	It is odd that the 'zero' value is assigned to pointer. Probably meant: *ptr = zero.	1
V528	It is odd that pointer is compared with the 'zero' value. Probably meant: *ptr != zero.	1
V535	The variable 'X' is being used for this loop and for the outer loop.	4
V541	It is dangerous to print the string into itself.	1
V547	Expression is always true/false.	18
V556	The values of different enum types are compared.	24
V560	A part of conditional expression is always true/false.	2
V571	Recurring check. This condition was already verified in previous line.	2
V572	It is odd that the object which was created using 'new' operator is immediately cast to another type.	4
V575	Function receives an odd argument.	82
V576	Incorrect format. Consider checking the N actual argument of the 'Foo' function.	5

V590	Consider inspecting this expression. The expression is excessive or contains a misprint.	3
V591	Non-void function should return a value.	3
V593	Consider reviewing the expression of the 'A = B == C' kind. The expression is calculated as following: 'A = (B == C)'.	1
V595	The pointer was utilized before it was verified against nullptr. Check lines: N1, N2.	28
V597	The compiler could delete the 'memset' function call, which is used to flush 'Foo' buffer. The RtlSecureZeroMemory() function should be used to erase the private data.	53
V601	An odd implicit type casting.	1
V609	Divide or mod by zero.	1
V610	Undefined behavior. Check the shift operator.	2
V611	The memory allocation and deallocation methods are incompatible.	2
V614	Uninitialized variable 'Foo' used.	1
V618	It's dangerous to call the 'Foo' function in such a manner, as the line being passed could contain format specification. The example of the safe code: printf("%s", str);	6
V622	Consider inspecting the 'switch' statement. It's possible that the first 'case' operator is missing.	1
V624	The constant NN is being utilized. The resulting value could be inaccurate. Consider using the M_NN constant from <math.h>.	2
V636	The expression was implicitly cast from integer type to real type. Consider utilizing an explicit type cast to avoid overflow or loss of a fractional part.	12
V640	The code's operational logic does not correspond with its formatting.	3
V642	Saving the function result inside the 'byte' type variable is inappropriate. The significant bits could be lost breaking the program's logic.	1
V645	The function call could lead to the buffer overflow. The bounds should not contain the size of the buffer, but a number of characters it can hold.	6
V646	Consider inspecting the application's logic. It's possible that 'else' keyword is missing.	4
V647	The value of 'A' type is assigned to the pointer of 'B' type.	1
V649	There are two 'if' statements with identical conditional expressions. The first 'if' statement contains function return. This means that the second 'if' statement is senseless.	1
V666	Consider inspecting NN argument of the function 'Foo'. It is possible that the value does not correspond with the length of a string which was passed with the YY argument.	6
V668	There is no sense in testing the pointer against null, as the memory was allocated using the 'new' operator. The exception will be generated in the case of memory allocation error.	55

V674	The expression contains a suspicious mix of integer and real types.	1
V675	Writing into the read-only memory.	1
V686	A pattern was detected: A (A && ...). The expression is excessive or contains a logical error.	2
V690	The class implements a copy constructor/operator=, but lacks the operator=/copy constructor.	8
V692	An inappropriate attempt to append a null character to a string. To determine the length of a string by 'strlen' function correctly, a string ending with a null terminator should be used in the first place.	2
V694	The condition (ptr - const_value) is only false if the value of a pointer equals a magic constant.	2
V696	The 'continue' operator will terminate 'do { ... } while (FALSE)' loop because the condition is always false.	2
V701	realloc() possible leak: when realloc() fails in allocating memory, original pointer is lost. Consider assigning realloc() to a temporary pointer.	111
V746	Type slicing. An exception should be caught by reference rather than by value.	32
V755	Copying from unsafe data source. Buffer overflow is possible.	1
V759	Violated order of exception handlers. Exception caught by handler for base class.	9
V760	Two identical text blocks detected. The second block starts with NN string.	1
V762	Consider inspecting virtual function arguments. See NN argument of function 'Foo' in derived class and base class.	6
V769	The pointer in the expression equals nullptr. The resulting value is meaningless and should not be used.	8
V773	The function was exited without releasing the pointer/handle. A memory/resource leak is possible.	6
V774	The pointer was used after the memory was released.	5
V778	Two similar code fragments were found. Perhaps, this is a typo and 'X' variable should be used instead of 'Y'.	2
V779	Unreachable code detected. It is possible that an error is present.	17
V780	The object of non-passive (non-PDS) type cannot be used with the function.	2
V783	Dereferencing of invalid iterator 'X' might take place.	4
V786	Assigning the value C to the X variable looks suspicious. The value range of the variable: [A, B].	1
Sum total:		909

- More than **2 400 000** code lines were analyzed (excluding comments).
- I detected **900** errors.
- On average, PVS-Studio detects **0.38 errors per 1000 lines** of code.
- If we proceed from the assumption that there are only 3 errors per 1000 lines of code, then we detect more than 10% of errors.

False positives weren't taken into account

- We didn't do even a minimal setting up of the analyzer, so there is no point in evaluating the percentage of false positives.
- Judging by my personal feelings, there aren't many false positives.
- The amount of false positives does not really matter, because if we start the cooperation, the false positives will be a headache of our team, not the Tizen developers.

- The whole Tizen project with the third-party libraries included is **72 500 000** lines of C, C++ code (excluding the comments).
- That means that I checked only **3.3%** of the code.
- Estimation:
We will be able to find and fix **27 000** errors in total.

PVS-Studio team is ready for the cooperation

- www.viva64.com
- support@viva64.com