



COREHARD



C++ COMMUNITY

**Информационная безопасность и разработка ПО:
про что стоит знать программисту**

Евгений Рыжков

Кто я?



Евгений Рыжков

- Руководитель и сооснователь PVS-Studio;
- PVS-Studio – статический анализатор кода для C, C++, C# и **Java**;
- Работаем на Windows, Linux и macOS.

viva64.com

Содержание

- **SAST**: Static Application Security Testing;
- **CWE**: Common Weakness Enumeration;
- **CVE**: Common Vulnerabilities and Exposures;
- **MISRA**: Motor Industry Software Reliability Association.
- SEI **CERT** Coding Standards.

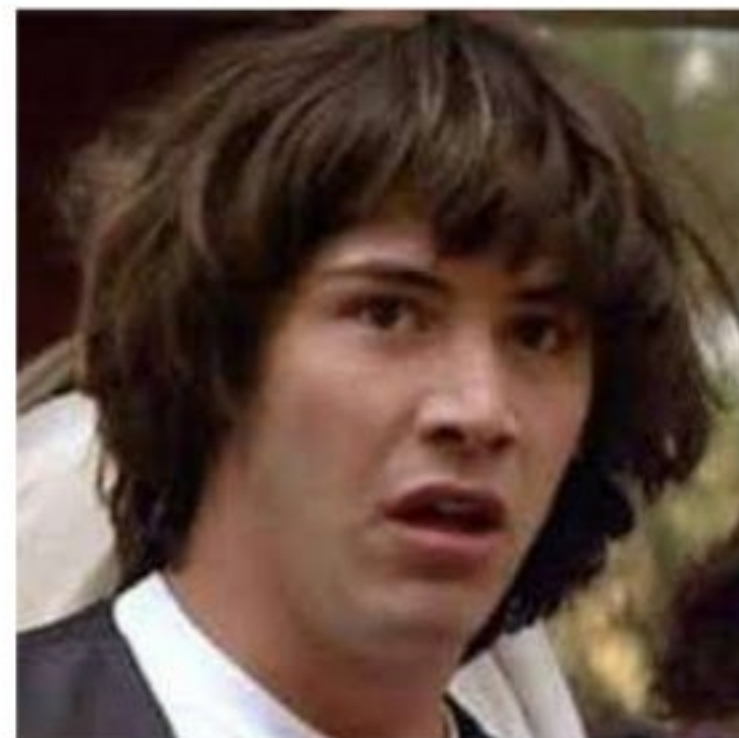


Зачем нам все это знать?

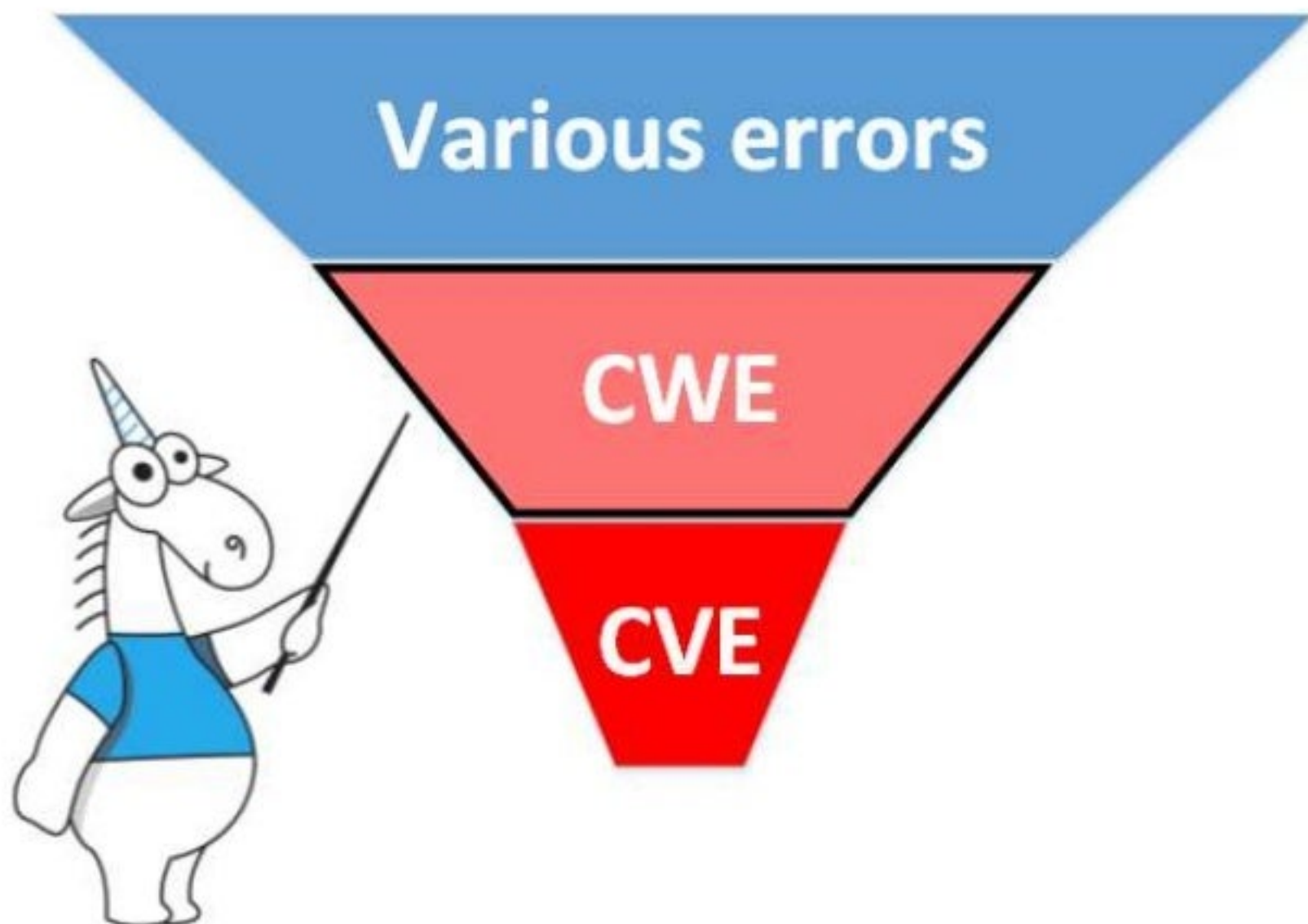
- Чтобы больше зарабатывать денег!
 - DevOps = системный администратор + программист
 - QA = тестировщик + программист
 - Security Researcher, Security Specialist, SecOps = security anything + программист

SAST: Static Application Security Testing

- Есть «обычный» статический анализ кода, который ищет «обычные» ошибки в программах.
- Кто бы мог подумать (сарказм), но проблемы безопасности и уязвимости в большинстве случаев происходят из-за ошибок в программах.
- Что если есть связь между ошибками и проблемами безопасности?



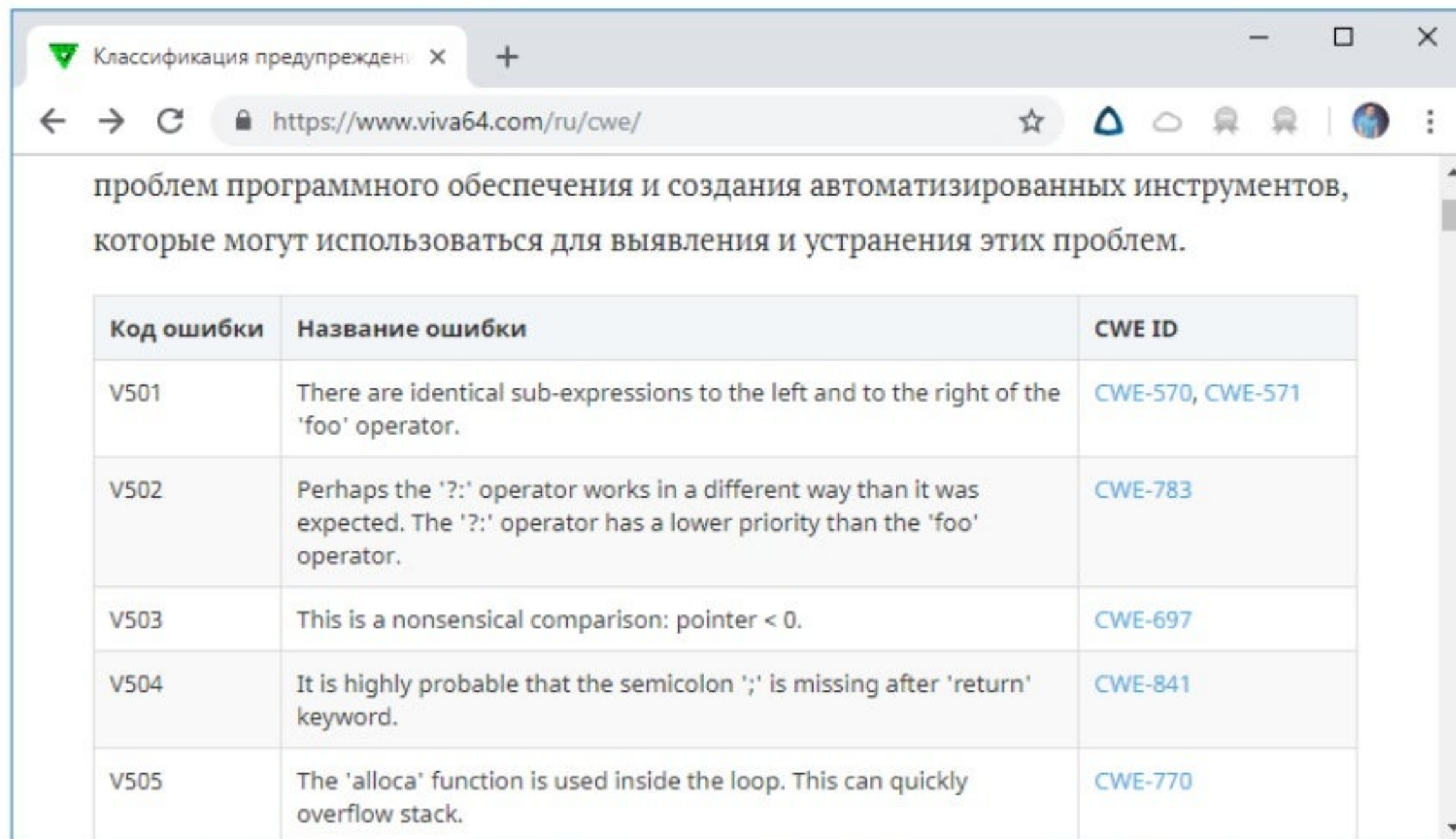
Связь и есть



- CWE – потенциальные уязвимости
- CVE – подтвержденные уязвимости

В PVS-Studio появилось соответствие диагностик анализатора и CWE

- Подробности: <https://www.viva64.com/ru/cwe/>



Код ошибки	Название ошибки	CWE ID
V501	There are identical sub-expressions to the left and to the right of the 'foo' operator.	CWE-570 , CWE-571
V502	Perhaps the '?:' operator works in a different way than it was expected. The '?:' operator has a lower priority than the 'foo' operator.	CWE-783
V503	This is a nonsensical comparison: pointer < 0.	CWE-697
V504	It is highly probable that the semicolon ';' is missing after 'return' keyword.	CWE-841
V505	The 'alloca' function is used inside the loop. This can quickly overflow stack.	CWE-770

Вот и весь SAST!

- Но давайте разбираться на примерах, что такое CWE и CVE.

CWE: Common Weakness Enumeration

- **CWE™** is a community-developed list of common software security weaknesses.
- Важно понимать, что это *потенциальные* уязвимости, которые могут стать реальными. А могут и не стать.
- Сайт: <https://cwe.mitre.org>
- CWE List Version 3.1 содержит 716 потенциальных уязвимостей.



Рассмотрим примеры

- CWE-570\571: Expression is Always False\True
- CWE-467: Use of sizeof() on a Pointer Type
- CWE-476: NULL Pointer Dereference
- CWE-14: Compiler Removal of Code to Clear Buffers
- CWE-369: Divide By Zero
- CWE-20: Improper Input Validation

CWE-570: Expression is Always False

```
#define BIT_READ  0x0001
#define BIT_WRITE 0x0010

unsigned int mask = BIT_READ & BIT_WRITE;

int hasReadWriteAccess(unsigned int userMask) {

    if (userMask & mask) {
        return 1;
    }

    return 0;
}
```


CWE-570: Expression is Always False


```
#define BIT_READ 0x0001
#define BIT_WRITE 0x0010

unsigned int mask = BIT_READ & BIT_WRITE; // 00000000

int hasReadWriteAccess(unsigned int userMask) {

    if (userMask & mask) { // always false
        return 1;
    }

    return 0;
}
```



CWE-14:

Compiler Removal of Code to Clear Buffers

```
void GetData(char *MFAddr) {  
    char pwd[64];  
    if (GetPasswordFromUser(pwd, sizeof(pwd))) {  
        if (ConnectToMainframe(MFAddr, pwd)) {  
            // Interaction with mainframe  
            ....  
        }  
    }  
    memset(pwd, 0, sizeof(pwd)); // <=  
}
```

CWE-14:

Compiler Removal of Code to Clear Buffers

```
int mlx5_core_create_qp(...)  
{  
    struct mlx5_destroy_qp_mbox_out dout;  
err_cmd:  
    ....  
    memset(&dout, 0, sizeof(dout)); // <=  
    ....  
}
```


CWE-20: Improper Input Validation

```
int c = getchar();
if (c < 0) {
    if (fgets(command_buf, sizeof(command_buf) - 1,
              stdin) != command_buf) {
        break;
    }
    command_buf[strlen(command_buf)-1] = '\0';
    break;
}
```

CWE-20: Improper Input Validation

```
int c = getchar();  
if (c < 0) {  
    if (fgets(command_buf, sizeof(command_buf) - 1,  
              stdin) != command_buf) {  
        break;  
    }  
    command_buf[strlen(command_buf)-1] = '\0';  
    break;  
}
```

CWE-20: Improper Input Validation

```
int c = getchar();
if (c < 0) {
    if (fgets(command_buf, sizeof(command_buf) - 1,
              stdin) != command_buf) {
        break;
    }
    command_buf[strlen(command_buf)-1] = '\0';
    break;
}
```


CWE-20: Improper Input Validation

```
int c = getchar();
if (c < 0) {
    if (fgets(command_buf, sizeof(command_buf) - 1,
              stdin) != command_buf) {
        break;
    }
    command_buf[0strlen(command_buf)-1] = '\0''\0';
    break;
}
```

CVE: Common Vulnerabilities and Exposures

- CVE - *реальные* уязвимости, найденные в приложениях.
- Основной сайт: <https://cve.mitre.org/>
- Total CVE Entries: 108 581
- Также интересно: <https://www.cvedetails.com/>
- Для маньяков: <https://twitter.com/CVEnew>




CVE-2017-15232

```
METHODDEF(void) quantize_ord_dither (.....,
                                       JSAMPARRAY output_buf,
                                       int num_rows) {
    .....
    for (row = 0; row < num_rows; row++) {
        jzero_far((void *) output_buf[row],
                  (size_t) (width * sizeof(JSAMPLE)));
        .....
    }
    .....
}
```


CVE-2017-15232

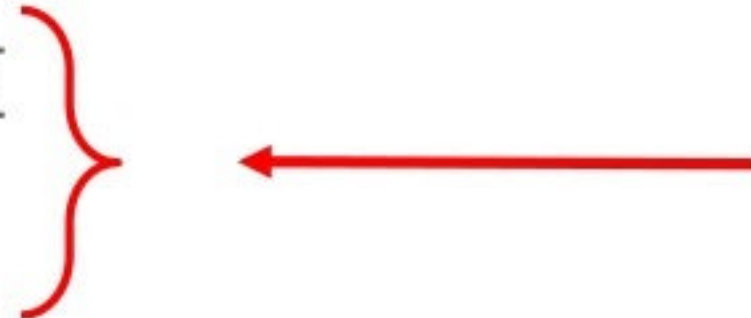
```
METHODDEF(void) quantize_ord_dither (.....,
                                       JSAMPARRAY output_buf,
                                       int num_rows) {
    ....
    for (row = 0; row < num_rows; row++) {
        jzero_far((void *) output_buf[row],           null
                  (size_t) (width * sizeof(JSAMPLE))); pointer
    }
    ....
}
```



CVE-2017-15232

```
METHODDEF(void) quantize_ord_dither (.....,
                                       JSAMPARRAY output_buf,
                                       int num_rows) {

    .....
    if (output_buf == NULL && num_rows) {
        ERREXIT(cinfo, JERR_BAD_PARAM);
    }
    for (row = 0; row < num_rows; row++) {
        jzero_far((void *) output_buf[row],
                  (size_t) (width * sizeof(JSAMPLE)));
    }
    .....
}
```




CVE-2015-8617

```
static void zend_throw_or_error(int fetch_type,
                                zend_class_entry *exception_ce,
                                const char *format, ...) {
    ....
    if (fetch_type & ZEND_FETCH_CLASS_EXCEPTION) {
        zend_throw_error(exception_ce, message);
    } else {
        zend_error(E_ERROR, "%s", message);
    }
    ....
}
```

CVE-2015-8617

```
static void zend_throw_or_error(int fetch_type,  
                                zend_class_entry *exception_ce,  
                                const char *format, ...) {  
  
    ....  
    if (fetch_type & ZEND_FETCH_CLASS_EXCEPTION) {  
        zend_throw_error(exception_ce, "%s", message);  
    } else {  
        zend_error(E_ERROR, "%s", message);  
    }  
  
    ....  
}
```

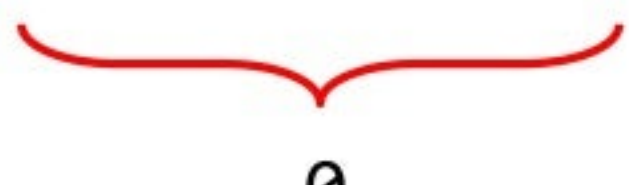


CVE-2017-15025

```
static .... decode_line_info (....) {  
    ....  
    case DW_LNS_const_add_pc:  
        if (lh.maximum_ops_per_insn == 1)  
            address += (lh.minimum_instruction_length  
                        * ((255 - lh.opcode_base) / lh.line_range));  
    ....  
}
```

CVE-2017-15025


```
static .... decode_line_info (....) {  
    ....  
    case DW_LNS_const_add_pc:  
        if (lh.maximum_ops_per_insn == 1)  
            address += (lh.minimum_instruction_length  
                        * ((255 - lh.opcode_base) / lh.line_range));  
    ....  
}
```



0

CVE-2017-15025

```
static .... decode_line_info (....) {  
    ....  
    case DW_LNS_const_add_pc:  
    if (lh.line_range == 0) }  
        goto line_fail; }  
    if (lh.maximum_ops_per_insn == 1)  
        address += (lh.minimum_instruction_length  
                    * ((255 - lh.opcode_base) / lh.line_range));  
    ....  
}
```



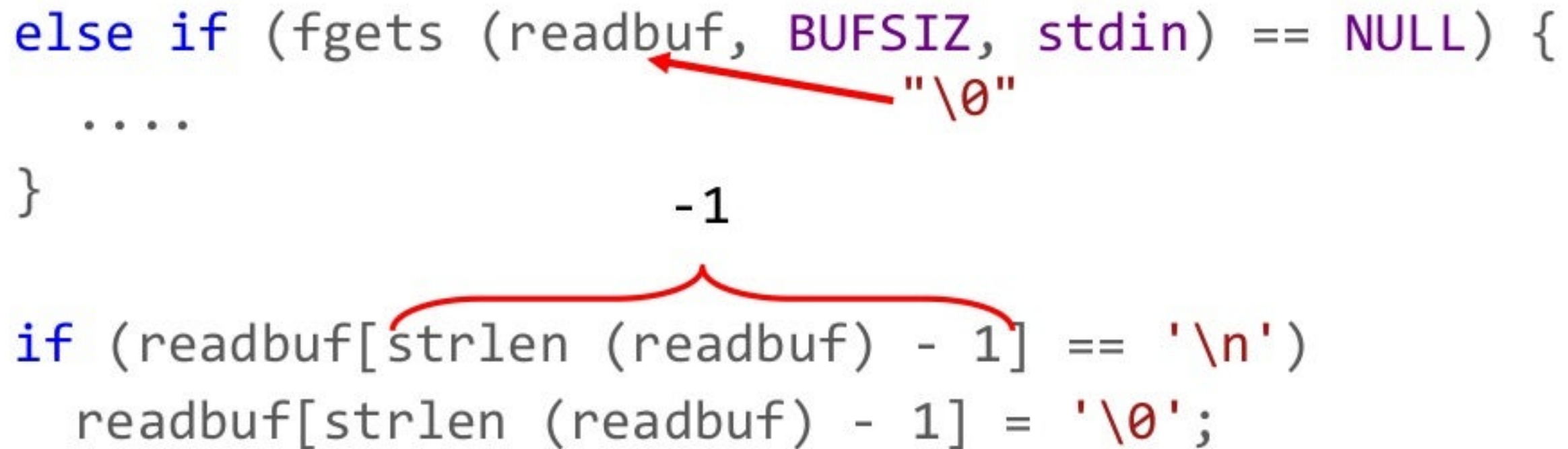
CVE-2015-8948

```
else if (fgets (readbuf, BUFSIZ, stdin) == NULL) {  
    ....  
}
```

```
if (readbuf[strlen (readbuf) - 1] == '\n')  
    readbuf[strlen (readbuf) - 1] = '\0';
```


CVE-2015-8948

```
else if (fgets (readbuf, BUFSIZ, stdin) == NULL) {  
    ....  
}  
-1  
if (readbuf[strlen (readbuf) - 1] == '\n')  
    readbuf[strlen (readbuf) - 1] = '\0';
```

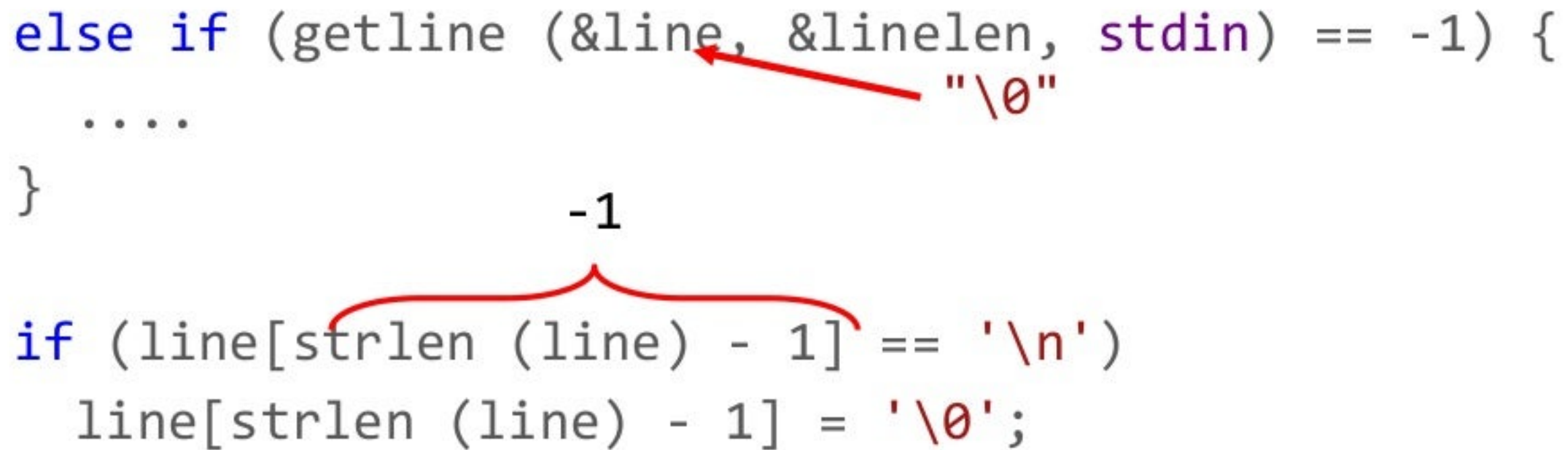


CVE-2016-6262

```
else if (getline (&line, &linelen, stdin) == -1) {  
    ...  
}  
  
if (line[strlen (line) - 1] == '\n')  
    line[strlen (line) - 1] = '\0';
```

CVE-2016-6262

```
else if (getline (&line, &linelen, stdin) == -1) {  
    ....  
}  
-1  
if (line[strlen (line) - 1] == '\n')  
    line[strlen (line) - 1] = '\0';
```



CVE-2016-6262

```
else if (getline (&line, &linelen, stdin) == -1) {  
    ....  
}  
  
if (strlen (line) > 0)  
    if (line[strlen (line) - 1] == '\n')  
        line[strlen (line) - 1] = '\0';
```


Два подхода к поиску уязвимостей

- Можно (и нужно) искать потенциальные уязвимости, которые могут привести к реальным проблемам при написании нового кода.
- Можно искать по базе существующих известных уязвимостей в своем имеющемся коде (с помощью автоматизированных средств).

MISRA C 2012 и MISRA C++ 2008

- **Закрытый** стандарт кодирования. Доступен только за деньги.
- Это не про ошибки, а про то как писать код так, чтобы ошибок и проблем в целом было поменьше.
- Анализаторы, проверяющие соответствие MISRA, не ищут ошибки.
- Стандарт состоит из правил, которые помогли бы писать более безопасный код. Хотя если взглянуть на правила первый раз, они могут показаться странными.

Примеры рекомендаций из MISRA

- Каждый 'switch' должен содержать 'default'.
- Возвращаемое значение non-void функций должно быть использовано.
- Функция должна иметь одну точку выхода.
- Указатели должны иметь не более 2-х уровней вложенности.
- Тела циклов и условных выражений должны быть заключены в {}.
- Каждый 'if ... else if' должен иметь завершающий 'else'.
- Не использовать unions.

The *goto* statement should not be used

```
for (x = 0; FEATURES[x].name; x++) {  
    if (all || !ns || !strcasecmp(ns, FEATURES[x].name)) {  
        if (!(tag = iks_insert (query, "feature")) {  
            goto fail; // <=  
        }  
        iks_insert_attrrib(tag, "var", FEATURES[x].name);  
    }  
}
```


All *if ... else if* constructs shall be terminated with an *else* statement

```
if (!strcasecmp(type, "unavailable")) {  
    dl_signal = LDL_SIGNAL_PRESENCE_OUT;  
} else if (!strcasecmp(type, "probe")) {  
    dl_signal = LDL_SIGNAL_PRESENCE_PROBE;  
}
```

All *if ... else if* constructs shall be terminated with an *else* statement

```
if (!strcasecmp(type, "unavailable")) {  
    dl_signal = LDL_SIGNAL_PRESENCE_OUT;  
} else if (!strcasecmp(type, "probe")) {  
    dl_signal = LDL_SIGNAL_PRESENCE_PROBE;  
} else {  
    // Do something...  
}
```

A function should have a single point of exit at the end

```
static iks* working_find(iks *tag, const char *name) {  
    while(tag) {  
        if (!strcasecmp(iks_name(tag), name)) {  
            return tag; // <=  
        }  
        tag = iks_next_tag(tag);  
    }  
    return NULL; // <=  
}
```

Dynamic heap memory allocation shall not be used

```
void clearSpline() {  
    for ( int i = 0; i < splinePoints.Num(); i++ ) {  
        delete splinePoints[i]; // <=  
    }  
    splinePoints.Clear();  
}
```


Octal constants shall not be used

```
if (!(ch & 0200)) { // <=
    ....
}
else {
    if ((ch & 0300) != 0300) { // <= x2
        ....
    }
    else {
        mask = 0340; // <=
        ....
    }
    ....
}
```

SEI CERT Coding Standard

- Carnegie Mellon University, Software Engineering Institute;
- CERT – computer emergency response team;
- Есть стандарты для C, C++, Java, Perl, Android.

EXP34-C. Do not dereference null pointers, C/C++

```
#include <png.h> /* From libpng */
#include <string.h>

void func(png_structp png_ptr, int length, const void *user_data)
{
    png_charp chunkdata;
    chunkdata = (png_charp)png_malloc(png_ptr, length + 1);
    /* ... */
    memcpy(chunkdata, user_data, length); // chunkdata may be NULL
    /* ... */
}
```

INT33-C. Ensure that division and remainder operations do not result in divide-by-zero errors

```
#include <limits.h>

void func(signed long s_a, signed long s_b) {
    signed long result;
    if ((s_a == LONG_MIN) && (s_b == -1)) {
        /* Handle error */
    } else {
        result = s_a / s_b; // <<
    }
    /* ... */
}
```


EXP12-C. Do not ignore values returned by functions

```
int main(int argc, char **argv)
{
    ....
    QByteArray arg(argv[a]);
    ....
    arg = arg.mid(1);
    arg.toLower(); // <<
    if (arg == "o")
        ....
}
```

Выводы

- В «безопасности» нет никакой магии.
- Но вы должны знать хотя бы ключевые слова и понимать о чем речь.
- Также полезно знать про инструменты, которые помогут искать проблемы безопасности в коде приложений.

COREHARD



+ Скачайте PVS-Studio и проверьте свой проект на программные ошибки и проблемы безопасности!

Языки: C, C++, C#, Java

Платформы: Windows, Linux, macOS

viva64.com