ГОСТ Р 71207–2024 — Статический анализ программного обеспечения. Критические ошибки

Вебинар от команды PVS-Studio

Андрей Карпов

DevRel

Андрей Карпов

- Один из основателей проекта PVS-Studio — pvs-studio.ru
- Пишу и рассказываю про статический анализ и качество кода

Критическая ошибка

- Ошибка, которая может привести к нарушению безопасности обрабатываемой информации.
- Мы в статьях называли такие ошибки потенциальными уязвимостями.

Интерпретируемые языки

- ошибки непроверенного использования чувствительных данных;
- ошибки некорректного использования системных процедур и интерфейсов, связанных с обеспечением информационной безопасности;
- ошибки при работе с многопоточными примитивами.

Компилируемые языки

- ошибки непроверенного использования чувствительных данных;
- ошибки некорректного использования системных процедур и интерфейсов, связанных с обеспечением информационной безопасности;
- ошибки при работе с многопоточными примитивами;
- ошибки целочисленного переполнения и некорректного совместного использования знаковых и беззнаковых чисел;
- ошибки переполнения буфера.

Дополнительно для C и C++

- ошибки разыменования нулевого указателя;
- ошибки деления на ноль;
- ошибки управления динамической памятью;
- ошибки использования форматной строки;
- ошибки использования неинициализированных переменных;
- ошибки утечек памяти, незакрытых файловых дескрипторов и дескрипторов сетевых соединений.

PVS-Studio и критические ошибки

- PVS-Studio умеет выявлять критические ошибки.
- Пока нет возможности выбрать подмножество критических ошибок.
- Сделаем.

Примеры критических ошибок в коде, которые должен находить анализатор

Ошибки непроверенного использования чувствительных данных (ReactOS, C)

```
int login(const char* host)
{
  char tmp[80];
  ....
  (void)fflush(stdout);
  (void)fgets(tmp, sizeof(tmp) - 1, stdin);
  tmp[strlen(tmp) - 1] = '\0';
  if (*tmp == '\0')
    user = myname;
  ....
}
```

PVS-Studio: V1010 Unchecked tainted data is used in index: 'strlen(tmp)'. ftp.c 216
https://pvs-studio.ru/ru/blog/posts/cpp/1122/

Другой пример из ONLYOFFICE Community Server, C#

```
public void SetCredentials(string userName, string password, string domain)
{
  if (string.IsNullOrEmpty(userName))
  {
    throw new ArgumentException("Empty user name.", "userName");
  }
  if (string.IsNullOrEmpty("password"))
  {
    throw new ArgumentException("Empty password.", "password");
  }
  ....
}
```

PVS-Studio: V3022 Expression 'string.IsNullOrEmpty("password")' is always false. SmtpSettings.cs 104

Ошибки некорректного использования системных процедур и интерфейсов, связанных с обеспечением ИБ (Crypto++, C++)

```
MicrosoftCryptoProvider::MicrosoftCryptoProvider()
{
  if(!CryptAcquireContext(&m_hProvider, 0, 0, PROV_RSA_FULL,
                          CRYPT_VERIFYCONTEXT))
    throw OS_RNG_Err("CryptAcquireContext");
}
```

PVS-Studio: V1109 The 'CryptAcquireContextA' function is deprecated. Consider switching to an equivalent newer function. osrng.cpp 50

> **Important** This API is deprecated. New and existing software should start using Cryptography Next Generation APIs. Microsoft may remove this API in future releases.

Ошибки некорректного использования системных процедур и интерфейсов, связанных с обеспечением ИБ (Android, C)

```
static void FwdLockGlue_InitializeRoundKeys()
{
  unsigned char keyEncryptionKey[KEY_SIZE];
  ....
  memset(keyEncryptionKey, 0, KEY_SIZE); // Zero out key data.
}
```

PVS-Studio: V597 The compiler could delete the 'memset' function call, which is used to flush 'keyEncryptionKey' buffer. The memset_s() function should be used to erase the private data. FwdLockGlue.c 102

Ошибки при работе с многопоточными примитивами (Zephyr, C)

```
static int nvs_startup(struct nvs_fs *fs) {
  ....
  k_mutex_lock(&fs->nvs_lock, K_FOREVER);
  ....
  if (fs->ate_wra == fs->data_wra && last_ate.len) {
    return -ESPIPE;
  }
  ....
end:
  k_mutex_unlock(&fs->nvs_lock);
  return rc;
}
```

PVS-Studio: V1020 The function exited without calling the 'k_mutex_unlock' function. Check lines: 620, 549. nvs.c 620

Ошибки при работе с многопоточными примитивами (WildFly, Java)

```
private volatile ExpressionFactory factory;
....
@Override
public ExpressionFactory getExpressionFactory() {
  if (factory == null) {
    synchronized (this) {
      if (factory == null) {
        factory = delegate.getExpressionFactory();
        for (ExpressionFactoryWrapper wrapper : wrapperList) {
          factory = wrapper.wrap(factory, servletContext);
        }
      }
    }
  }
  return factory;
}
```

PVS-Studio: V6082 Unsafe double-checked locking. A previously assigned object may be replaced by another object. JspApplicationContextWrapper.java(74), JspApplicationContextWrapper.java(72)

Целочисленное переполнение (C++)

```
int foo(const unsigned char *s)
{
  int r = 0;
  while(*s) {
    r += ((r * 20891 + *s *200) | *s ^ 4 | *s ^ 3) ^ (r >> 1);
    s++;
  }
  return r & 0x7fffffff;
}
```

PVS-Studio: V1026. The variable is incremented in the loop. Undefined behavior will occur in case of signed integer overflow.
https://pvs-studio.ru/ru/docs/warnings/v1026/

Целочисленное переполнение (libtorrent, C++)

```
void torrent::get_download_queue(std::vector<partial_piece_info>* queue) const
{
  ....
  const int blocks_per_piece = m_picker->blocks_in_piece(piece_index_t(0));
  ....
  int counter = 0;
  for (auto i = q.begin(); i != q.end(); ++i, ++counter)
  {
    ....
    pi.blocks = &blk[std::size_t(counter * blocks_per_piece)];
  }
```

PVS-Studio: V1028 Possible overflow. Consider casting operands of the 'counter * blocks_per_piece' operator to the 'size_t' type, not the result. torrent.cpp 7092

Некорректное совместное использование знаковых и беззнаковых чисел (Hypertext Preprocessor, C)

```
PHP_CLI_API size_t sapi_cli_single_write(....)
{
  ....
  size_t shell_wrote;
  shell_wrote = cli_shell_callbacks.cli_shell_write(....);
  if (shell_wrote > -1) {
    return shell_wrote;
  }
  ....
}
```

PVS-Studio: V605 Consider verifying the expression: shell_wrote > - 1. An unsigned value is compared to the number -1. php_cli.c 266

Ошибки переполнения буфера (GPCS4, C++)

```
struct GnmCmdPSShader {
  ....
  uint32_t reserved[27];
};

int PS4API sceGnmSetPsShader350(....) {
  ....
  memset(param->reserved, 0, sizeof(param->reserved) * sizeof(uint32_t));
  return SCE_OK;
}
```

PVS-Studio:

- V531 It is odd that a sizeof() operator is multiplied by sizeof(). sce_gnm_draw.cpp 420
- V512 A call of the 'memset' function will lead to overflow of the buffer 'param->reserved'. sce_gnm_draw.cpp 420

Ошибки переполнения буфера (ICU, C)

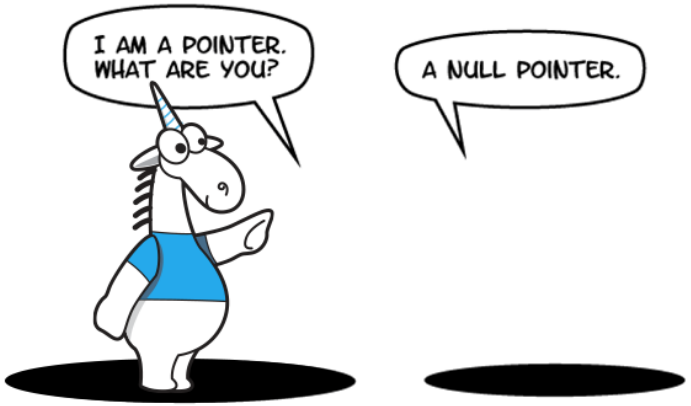
```
static char plugin_file[2048] = "";
U_CAPI void U_EXPORT2 uplug_init(UErrorCode *status) {
  ....
  // uprv_strncat раскрывается в strncat
  uprv_strncpy(plugin_file, plugin_dir,          2047);
  uprv_strncat(plugin_file, U_FILE_SEP_STRING,   2047);
  uprv_strncat(plugin_file, "icuplugins",        2047);
  uprv_strncat(plugin_file, U_ICU_VERSION_SHORT, 2047);
  uprv_strncat(plugin_file, ".txt",              2047);
  ....
```

Свободное место в буфере, а не размер буфера!

PVS-Studio: V645 The 'strncat' function call could lead to the 'plugin_file' buffer overflow. The bounds should not contain the size of the buffer, but a number of characters it can hold. icuplug.c

Ошибки разыменования нулевого указателя (OpenJDK, C)

```
static jint JNICALL cbObjectTagInstance(....)
{
  ClassInstancesData  *data;
  data = (ClassInstancesData*)user_data;
  if (data == NULL) {
    data->error = AGENT_ERROR_ILLEGAL_ARGUMENT;
    return JVMTI_VISIT_ABORT;
  }
  ....
}
```



PVS-Studio: V522 Dereferencing of the null pointer 'data' might take place. util.c 2424

Ошибки деления на ноль (LLVM, C)

```
COMPILER_RT_ABI du_int __udivmoddi4(...., du_int* rem) {
  ....
  if (d.s.low == 0) {
    if (d.s.high == 0) {
      if (rem)
        *rem = n.s.high % d.s.low;
      return n.s.high / d.s.low;
    }
```

PVS-Studio:

- V609 Mod by zero. Denominator 'd.s.low' == 0. udivmoddi4.c 61
- V609 Divide by zero. Denominator 'd.s.low' == 0. udivmoddi4.c 62

Ошибки управления динамической памятью (OpenToonz, C++)

```
template <class T> void doDirectionalBlur(....) {
  T *row, *buffer;
  ....
  row = new T[lx + 2 * brad + 2];
  ....
  free(row);
  r->unlock();
}
```

PVS-Studio: V611 The memory was allocated using 'new' operator but was released using the 'free' function. Consider inspecting operation logics behind the 'row' variable. motionblurfx.cpp 288

Ошибки управления динамической памятью (MuseScore, C++)

```
void GuitarPro6::readGpif(QByteArray* data)
{
  ....
  } else {
    delete slur;
    legatos[slur->track()] = 0;
  }
```

PVS-Studio: V774 The 'slur' pointer was used after the memory was released. importgtp-gp6.cpp 2592

Ошибки использования форматной строки (OpenCOLLADA, C++)

```
struct vector
{
  double x;
  double y;
  double z;
  void write(FILE* file) const
  {
    fprintf(file, "%f %f %f %f", x, y, z);
  }
```

PVS-Studio: V576 Incorrect format. A different number of actual arguments is expected while calling 'fprintf' function. Expected: 6. Present: 5. mayadmtypes.h 657

Ошибки использования форматной строки (WinSCP, C++)

```
bool CAsyncSslSocketLayer::CreateSslCertificate(....)
{
  ....
  char buffer[1001];
  int len;
  while ((len = pBIO_read(bio, buffer, 1000)) > 0)
  {
    buffer[len] = 0;
    fprintf(file, buffer);
  }
  ....
}
```

Это ещё и ошибка непроверенного использования чувствительных данных.

PVS-Studio: V618 It's dangerous to call the 'fprintf' function in such a manner, as the line being passed could contain format specification. The example of the safe code: printf("%s", str); asyncsslsocketlayer.cpp 2247

Использование неинициализированных переменных (CovidSim Model, C++)

```
void CalcLikelihood(....) {
  ....
  double ModelValue;
  // loop over all days of infection up to day of sample
  for (int k = offset; k < day; k++)
  {
    // add P1 to P2 to prevent degeneracy
    double prob_seroconvert = P.SeroConvMaxSens * ....;
    ModelValue += c * TimeSeries[k - offset].incI * prob_seroconvert;
  }
```

PVS-Studio: V614 Uninitialized variable 'ModelValue' used. CovidSim.cpp 5412

Ошибки утечек памяти (PMDK, C)

```
static enum pocli_ret
pocli_args_obj_root(...., char* in, PMEMoid** oidp)
{
    char* input = strdup(in);
    if (!input)
        return POCLI_ERR_MALLOC;

    if (!oidp)
        return POCLI_ERR_PARS;
```

PVS-Studio: V773 The function was exited without releasing the 'input' pointer. A memory leak is possible. pmemobjcli.c 238

И других ресурсов (CMake, C)

```
RHASH_API int rhash_file(....)
{
    FILE* fd;
    ....
    if ((fd = fopen(filepath, "rb")) == NULL) return -1;
    if ((ctx = rhash_init(hash_id)) == NULL) return -1;
    res = rhash_file_update(ctx, fd);
    fclose(fd);
```

PVS-Studio: V773 The function was exited without closing the file referenced by the 'fd' handle. A resource leak is possible. rhash.c 450

Комментарии

Критические ошибки можно и нужно править без учёта, уязвимость это или нет

- В настоящем стандарте в задачи статического анализа **не входит разграничение ошибок в части последствий**, необходимо найти потенциальные места ошибок.

- Мы часто разбирали это в статьях. Приятно, что это звучит и в стандарте.

«Некритическая ошибка» тоже может быть опасной

- К классификации следует относиться вдумчиво.
- Всё условно и относительно.

- Формально верно найденная критическая ошибка может не представлять угрозы.
- Ошибка, не классифицированная как критическая, вполне может такой быть.

Заключение

- Следующий вебинар как раз будет посвящён технологиям анализа кода.
- Пишите нам, если хотите узнать про какие-то темы подробнее. Мы собираем идеи для будущих вебинаров.
- Подписывайтесь и общайтесь с нами в VK:
  - https://vk.com/**pvsstudio_rus**

Q&A

Андрей Карпов

DevRel