На что нужно обратить внимание при обзоре кода разрабатываемой библиотеки

Андрей Карпов

CTO, PVS-Studio

karpov@viva64.com

www.viva64.com

Код, который мы обсуждаем на C++Russia

Реальный код

Девиз доклада: сделаем код лучше!

C++

О докладчике

- Карпов Андрей Николаевич
- Присутствую на Habr под именем Andrey2008 habr.com/users/andrey2008/
- Технический директор ООО «СиПроВер»
- Microsoft MVP
- Intel Black Belt Software Developer
- Один из основателей проекта PVS-Studio

Главное

- Разработчики библиотек должны быть **аккуратнее**, чем «классические» прикладные программисты

Почему?

- Неизвестно, где и как будет использоваться библиотека
- Платформы
- Компиляторы
- Оптимизации
- Сценарии использования

Прощай, проблема 2000 года

Здравствуй, проблема 2038 года

Продолжает использоваться 32-битный тип **time_t**

2038

- Особенно актуально:
  - для встраиваемых систем
  - для библиотек

- Джонатан Корбет. 2038: остался всего ~~21 год~~ 19 лет. https://www.viva64.com/ru/b/0505/

#pragma warning(default: NNN)

- Некорректный паттерн:

```
#pragma warning(disable:12345)
... код ...
#pragma warning(default:12345)
```

Рассмотрим на примере C4201

- C4201: безымянные структуры или объединения. **Уровень 4**.
- Выдаётся при **/Wall**.
- Если написать **default**, то предупреждение перестанет выдаваться, так как по умолчанию оно является отключенным.

```
// d3dtypes.h
// Direct3D types include file

#pragma warning(disable:4201) // anonymous unions warning
// .....
typedef struct _D3DCOLORVALUE {
  union {
    D3DVALUE r;
    D3DVALUE dvR;
  };
  //....
} D3DCOLORVALUE;
// ....
#pragma warning(default:4201)
```

Правильно: push, pop

```
#pragma warning(push)
#pragma warning(disable: NNNN)
....
// Корректный код, на который выдаётся предупреждение NNNN
....
#pragma warning(pop)
```

#pragma warning(default: NNN)

- Не страшно в вашем *.cpp файле
- Недопустимо в заголовочном файле библиотеки:
  - В коде пользователя могут исчезнуть полезные предупреждения
  - В коде пользователя могут появиться лишние предупреждения

64-битные ошибки

- Много раз писал и рассказывал
- Подробнее:

  - Разработка 64-битных приложений на языке C, C++
    https://www.viva64.com/ru/l/full/

  - DevGAMM 2018. Паттерны 64-битных ошибок в играх
    https://youtu.be/uzmdelgQYuc

64-битные ошибки

- Важно: то, что «прокатит» в вашем коде, неприемлемо для библиотеки

```
void Init(double *array, size_t n)
{
  for (int i = 0; i < n; i++)
    array[i] = 1.0;
}
```

Минутка 64-битного юмора

- StackOverflow:
  C and static Code analysis: Is this safer than memcpy?
  https://stackoverflow.com/questions/55239222/c-and-static-code-analysis-is-this-safer-than-memcpy

- Анализатор выдаёт предупреждение на функцию **memset**;
- «Решим проблему» написанием собственной функции копирования!

Минутка 64-битного юмора

```
void myMemCpy(void *dest, void *src, size_t n)
{
  char *csrc = (char *)src;
  char *cdest = (char *)dest;

  for (int i=0; i<n; i++)
    cdest[i] = csrc[i];
}
```

exit, abort, terminate

- Нельзя просто так взять и завершить работу приложения внутри библиотеки!

malloc, realloc

Разыменовывание нулевого указателя = UB

- Нельзя надеяться, что сразу будет именно падение;
- Можно успеть испортить данные, обрабатываемые в параллельных потоках.

```
void *memset(void *dest, int c, size_t n)
{
  size_t k;
  if (!n) return dest;
  dest[0] = dest[n-1] = c;
  if (n <= 2) return dest;
  //....
}
```

А кто сказал, что будет нулевое смещение?

```
static void
st_collections_group_parts_part_description_filter_data(void)
{
  //....
  filter->data_count++;
  array = realloc(filter->data,
    sizeof(Edje_Part_Description_Spec_Filter_Data) *
    filter->data_count);
  array[filter->data_count - 1].name = name;
  array[filter->data_count - 1].value = value;
  filter->data = array;
}
```

Библиотека EFL

Разыменовывание нулевого указателя - это потенциальная уязвимость

- Упасть при нехватке памяти - это **отказ в обслуживании**;
- Коду библиотек недопустимо так себя вести;
- Следует выбросить исключение или вернуть статус ошибки.

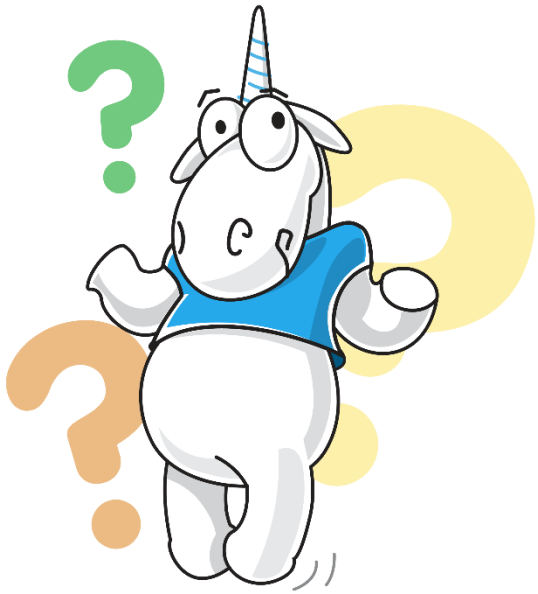
xmalloc для библиотек - это не решение

- xmalloc завершает работу приложения, если не удалось выделить память;
- Можно вызывать в своём коде;
- Вызывать в коде библиотек **НЕЛЬЗЯ**.

Неопределённое поведение

- У кода библиотек больше шансов, что себя негативно проявит неопределённое поведение
- Рассмотрим интересную функцию, которая перестала работать из-за «глючного GCC 8.0» https://www.linux.org.ru/forum/development/14422428

```
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;
}
```

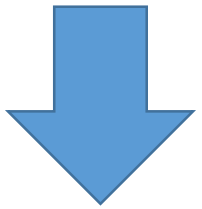


Опасное использование printf

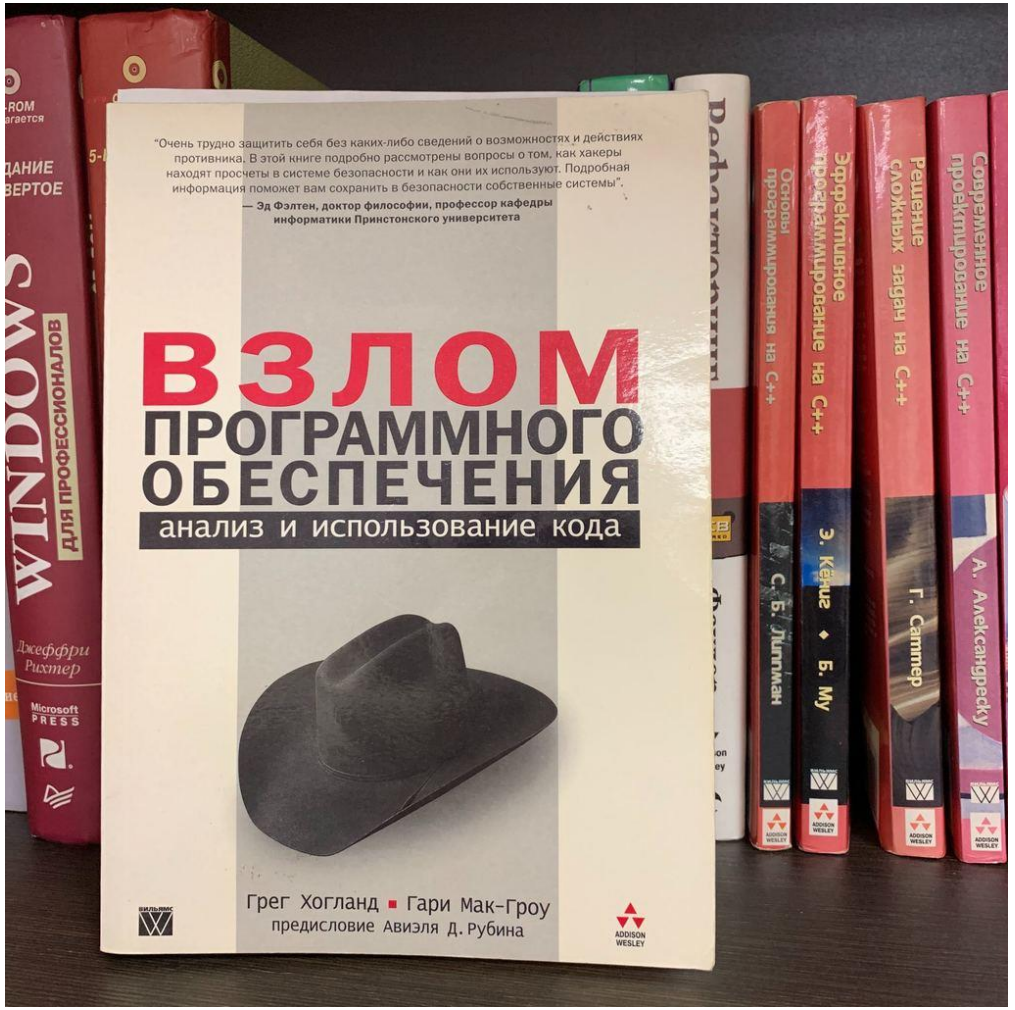
```
char buffer[1001];
int len;
while ((len = pBIO_read(bio, buffer, 1000)) > 0)
{
  buffer[len] = 0;
  fprintf(file, buffer);
}
```

Опасное использование printf

```
char *p;
//....
printf(p);
```



```
char *p;
//....
printf("%s", p);
```



Не создавайте макросы/функции с именами, совпадающими со стандартными

- Первый пример кода, не из библиотеки, но демонстрирует суть.

```
static int EatWhitespace (FILE * InFile)
{
  int c;
  for (c = getc (InFile);
       isspace (c) && ('\n' != c);
       c = getc (InFile))
    ;
  return (c);
}
```

Midnight Commander.
Где ошибка?

Ошибку не найти, пока не заглянешь в заголовочный файл

```
#ifdef isspace
#undef isspace
#endif
....
#define isspace(c) ((c)==' ' || (c) == '\t')
```

А что не так с этим кодом?

```
bool set_value_convert(char_t*& dest, uintptr_t& header,
                       uintptr_t header_mask, int value)
{
  char buf[128];
  sprintf(buf, "%d", value);
  return set_value_buffer(dest, header, header_mask, buf);
}
```

Библиотека StarEngine: definesTypes.h

```
#define sifstream std::ifstream
#define sfstream std::fstream
#define schar char
#define suchar unsigned schar
#define sprintf std::printf
#define satof atof
#define satoi atoi
#define sstrlen strlen
```

Неинициализированные данные

NCBI Genome Workbench

```
class CSnpBitAttributes
{
  // ....
  Uint8 m_BitSet;
};

inline CSnpBitAttributes::CSnpBitAttributes(
  const vector<char>& octet_string)
{
  auto count = sizeof(m_BitSet);
  auto byte = octet_string.end();
  do
    m_BitSet = (m_BitSet << 8) | *--byte;
  while (--count > 0);
}
```

Неинициализированные данные

- В библиотеках имеет смысл быть большим параноиком и инициализировать про запас.

Что дозволено Юпитеру, не дозволено быку

- Помните, что отныне **this != nullptr**.

```
void CWnd::AttachControlSite(CHandleMap* pMap)
{
  if (this != NULL && m_pCtrlSite == NULL)
  {
    //....
  }
}
```

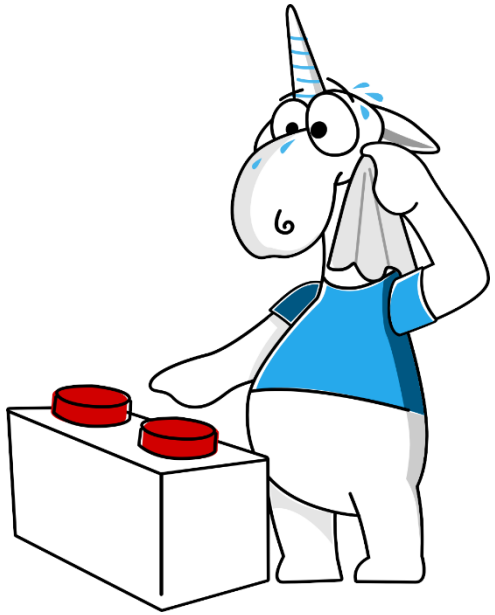
Занимательный факт

- Во времена C++ 1983 (Cfront) можно было делать так:

```
this = p;
```

Чётко различайте в интерфейсах BSTR и wchar_t *

- Иногда забывают о различиях между BSTR и wchar_t *;
- И в интерфейсах используются неверные типы со всеми вытекающими последствиями.



BSTR и wchar_t *

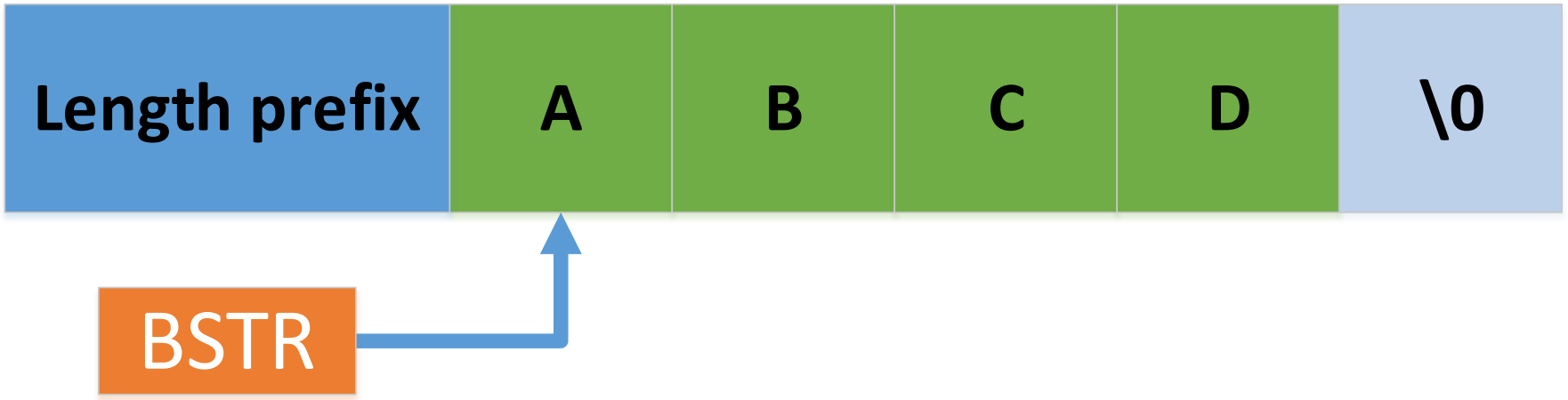
- wchar_t * – нуль-терминированная строка;
- BSTR – это строковый тип данных, который используется в COM, Automation и Interop функциях;
- «Разный физический смысл», но с точки зрения C и C++ это одинаковые типы:

```
typedef wchar_t OLECHAR;
typedef OLECHAR * BSTR;
```

BSTR

- Префикс длины (4 байта);
- Строка символов в кодировке Unicode;
- Терминальный ноль;

- Тип BSTR является указателем, который указывает на первый символ строки, а не на префикс длины.



BSTR

- Неправильно:

```
BSTR MyBstr = L"I am a happy BSTR";
```

- Правильно:

```
BSTR MyBstr = SysAllocString(L"I am a happy BSTR");
```

BSTR

- Некорректно работать с BSTR, как с wchar_t *;
- Чётко различайте эти типы в интерфейсах;
- Не вводите пользователей в заблуждение.

```
BSTR str = foo();
str += 3; // Теперь str это испорченная BSTR строка
```

Внимательно относитесь к усечению данных

- Классические варианты потери значимых бит:

```
int i = fgetc(stream);
char c = i;

void *ptr = foo();
int n = (int)ptr;
ptr = (void *)n;
```

Усечение длины строк

```
std::string str(foo());

short len = str.length();
```

Открывает путь потенциальным уязвимостям. Непростительно для библиотек.

```
size_t len = str.length();

std::string::size_type len = str.length();

auto len = str.length();
```

Усечение: memcmp (CVE-2012-2122)

MySQL

```
typedef char my_bool;

my_bool
check_scramble(const char *scramble_arg, const char *message,
               const uint8 *hash_stage2)
{
  ....
  return memcmp(hash_stage2, hash_stage2_reassured,
                SHA1_HASH_SIZE);
}
```

Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ
0	0	спец. NOP	32	20	сцеп. SP (Пробел)	64	40	@	96	60	`	128	80	Ђ	160	A0		192	C0	А	224	E0	а
1	1	спец. SOH	33	21	!	65	41	A	97	61	a	129	81	Ѓ	161	A1	Ў	193	C1	Б	225	E1	б
2	2	спец. STX	34	22	"	66	42	B	98	62	b	130	82	‚	162	A2	ў	194	C2	В	226	E2	в
3	3	спец. ETX	35	23	#	67	43	C	99	63	c	131	83	ѓ	163	A3	Ј	195	C3	Г	227	E3	г
4	4	спец. EOT	36	24	$	68	44	D	100	64	d	132	84	„	164	A4	¤	196	C4	Д	228	E4	д
5	5	спец. ENQ	37	25	%	69	45	E	101	65	e	133	85	…	165	A5	Ґ	197	C5	Е	229	E5	е
6	6	спец. ACK	38	26	&	70	46	F	102	66	f	134	86	†	166	A6	¦	198	C6	Ж	230	E6	ж
7	7	спец. BEL	39	27	'	71	47	G	103	67	g	135	87	‡	167	A7	§	199	C7	З	231	E7	з
8	8	спец. BS	40	28	(	72	48	H	104	68	h	136	88	€	168	A8	Ё	200	C8	И	232	E8	и
9	9	спец. Табуляция	41	29	)	73	49	I	105	69	i	137	89	‰	169	A9	©	201	C9	Й	233	E9	й
10	0A	спец. LF (Возвр. каретки)	42	2A	*	74	4A	J	106	6A	j	138	8A	Љ	170	AA	Є	202	CA	К	234	EA	к
11	0B	спец. VT	43	2B	+	75	4B	K	107	6B	k	139	8B	‹	171	AB	«	203	CB	Л	235	EB	л
12	0C	спец. FF	44	2C	,	76	4C	L	108	6C	l	140	8C	Њ	172	AC	¬	204	CC	М	236	EC	м
13	0D	спец. CR (Новая строка)	45	2D	-	77	4D	M	109	6D	m	141	8D	Ќ	173	AD	-	205	CD	Н	237	ED	н
14	0E	спец. SO	46	2E	.	78	4E	N	110	6E	n	142	8E	Ћ	174	AE	®	206	CE	О	238	EE	о
15	0F	спец. SI	47	2F	/	79	4F	O	111	6F	o	143	8F	Џ	175	AF	Ї	207	CF	П	239	EF	п
16	10	спец. DLE	48	30	0	80	50	P	112	70	p	144	90	ђ	176	B0	°	208	D0	Р	240	F0	р
17	11	спец. DC1	49	31	1	81	51	Q	113	71	q	145	91	'	177	B1	±	209	D1	С	241	F1	с
18	12	спец. DC2	50	32	2	82	52	R	114	72	r	146	92	'	178	B2	І	210	D2	Т	242	F2	т
19	13	спец. DC3	51	33	3	83	53	S	115	73	s	147	93	“	179	B3	і	211	D3	У	243	F3	у
20	14	спец. DC4	52	34	4	84	54	T	116	74	t	148	94	”	180	B4	ґ	212	D4	Ф	244	F4	ф
21	15	спец. NAK	53	35	5	85	55	U	117	75	u	149	95	•	181	B5	µ	213	D5	Х	245	F5	х
22	16	спец. SYN	54	36	6	86	56	V	118	76	v	150	96	–	182	B6	¶	214	D6	Ц	246	F6	ц
23	17	спец. ETB	55	37	7	87	57	W	119	77	w	151	97	—	183	B7	·	215	D7	Ч	247	F7	ч
24	18	спец. CAN	56	38	8	88	58	X	120	78	x	152	98	￿	184	B8	ё	216	D8	Ш	248	F8	ш
25	19	спец. EM	57	39	9	89	59	Y	121	79	y	153	99	™	185	B9	№	217	D9	Щ	249	F9	щ
26	1A	спец. SUB	58	3A	:	90	5A	Z	122	7A	z	154	9A	љ	186	BA	є	218	DA	Ъ	250	FA	ъ
27	1B	спец. ESC	59	3B	;	91	5B	[	123	7B	{	155	9B	›	187	BB	»	219	DB	Ы	251	FB	ы
28	1C	спец. FS	60	3C	<	92	5C	\	124	7C	|	156	9C	њ	188	BC	j	220	DC	Ь	252	FC	ь
29	1D	спец. GS	61	3D	=	93	5D	]	125	7D	}	157	9D	ќ	189	BD	S	221	DD	Э	253	FD	э
30	1E	спец. RS	62	3E	>	94	5E	^	126	7E	~	158	9E	ћ	190	BE	s	222	DE	Ю	254	FE	ю
31	1F	спец. US	63	3F	?	95	5F	_	127	7F		159	9F	џ	191	BF	ї	223	DF	Я	255	FF	я

Windows кодировка **CP1251**.

Буква 'я' имеет код 255.

Усечение: EOF

Android, C

```
char c;
printf("%s is already in *.base_fs format, just copying ....);
rewind(blk_alloc_file);
while ((c = fgetc(blk_alloc_file)) != EOF) {
  fputc(c, base_fs_file);
}
```

И ещё про EOF

```
while (!cin.eof())
  cin >> x;

while (!cin.eof() && !cin.fail())
  cin >> x;

while (cin)
  cin >> x;
```

memset и CWE-14

```
EAPI Eina_Binbuf *emile_binbuf_decipher(....)
{
   ....
   unsigned char ik[MAX_KEY_LEN];
   unsigned char iv[MAX_IV_LEN];
   ....
on_error:
   memset(iv, 0, sizeof (iv));
   memset(ik, 0, sizeof (ik));
   return NULL;
}
```

EFL Core Libraries

memset и CWE-14

- Безопасная очистка приватных данных
  https://www.viva64.com/ru/b/0388/

- CWE-14: Compiler Removal of Code to Clear Buffers
  https://cwe.mitre.org/data/definitions/14.html


Ты
ЗАЩИТИЛ СВОЙ КОД
ОТ БАГОВ?

Быстрая реакция на уязвимости

- От библиотеки зависит сразу множество приложений;
- Часто покупают ответственность, а не функционал;
- Не стесняйтесь брать за это деньги;
- Продаются дорогие платные библиотеки для создания PDF, хотя полно бесплатных;

- Берите деньги и делайте хорошо!

Code Review

- Обработчики ошибок (обычно плохо тестируются);
  - корректность;
  - нет exit, abort и т.д.
- malloc, realloc;
- Именования функций и макросов;
  - неспроста придумали namespace;
- Особенная внимательность к UB;
- Особенная внимательность к усечению старших бит;
- Инициализация и очистка памяти.

Рекомендации

- Уделять больше внимания Code Review;
- Использовать статические анализаторы кода;
- Google открыл систему для создания sandbox-окружений для библиотек C/C++: https://www.opennet.ru/opennews/art.shtml?num=50349

Ответы на вопросы

- Андрей Карпов
- CTO, PVS-Studio, www.viva64.com
- E-Mail: karpov@viva64.com
- Twitter: @Code_Analysis
- LinkedIn: andrey-karpov-pvs-studio