Software Engineering Conference Russia 2018

October 12-13
Moscow

Статический анализ кода: от опечаток к уязвимостям

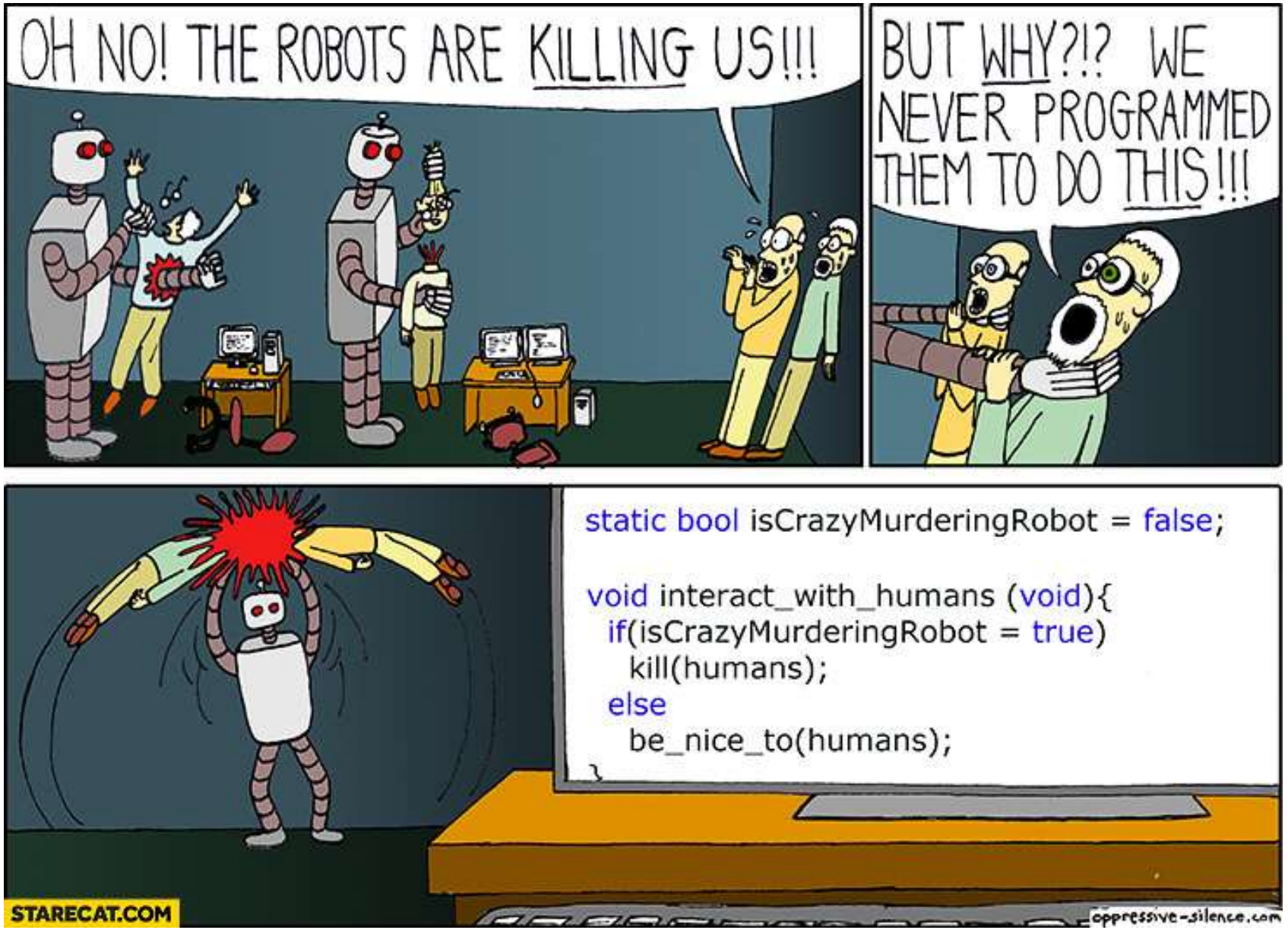
Сергей Хренов

PVS-Studio

«Одинаковые ошибки необязательно делать каждый раз, достаточно сделать одну, а затем обращаться к ней по мере необходимости из любого места программы».

Программистские байки

Почему полезен статический анализ кода

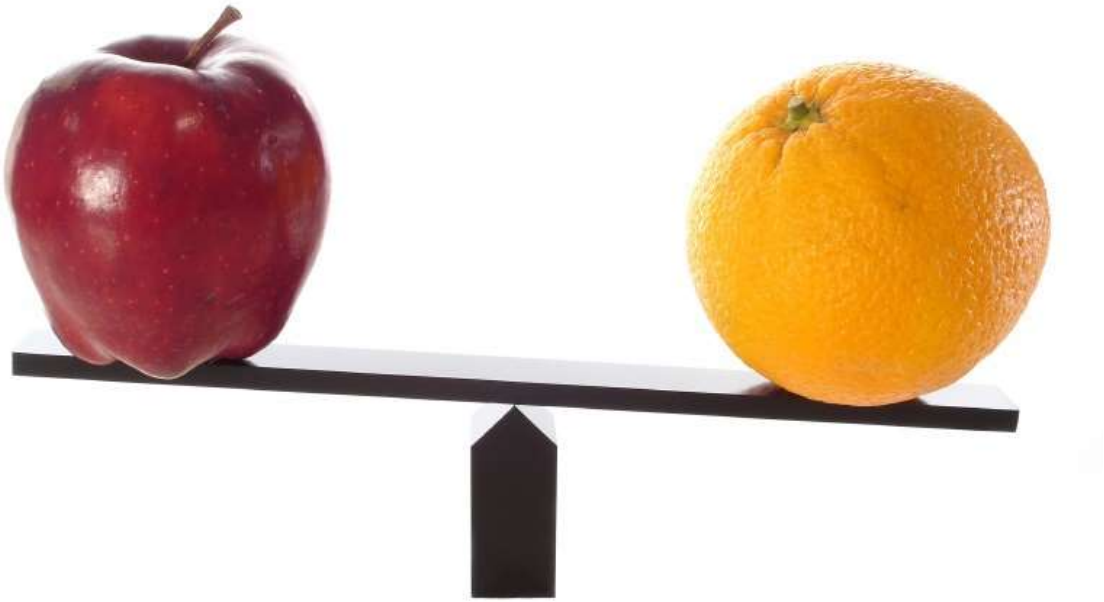


SAST - Static Application Security Testing

- Статический анализ, нацеленный на поиск и предотвращение уязвимостей.
- Уязвимости - те же самые обыкновенные ошибки.
- Всё больше внимания уделяется безопасности ПО.
- Актуальность SAST непрерывно растёт.
- PROFIT!

Выбирай любой

- PVS-Studio
- Coverity
- Klocwork
- ReSharper
- SonarQube
- Visual Studio
- Clang
- GCC
- Etc...

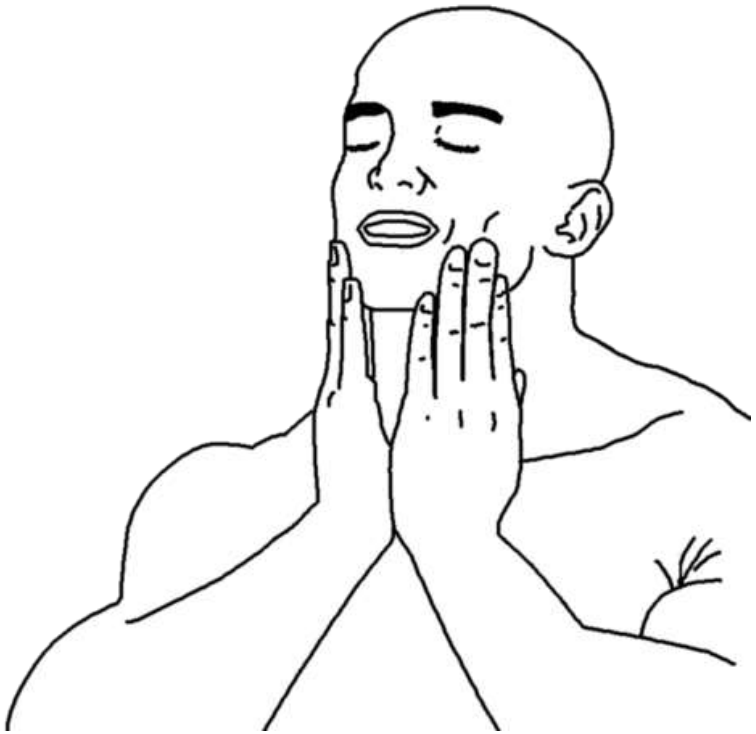


Как всё сделать не так

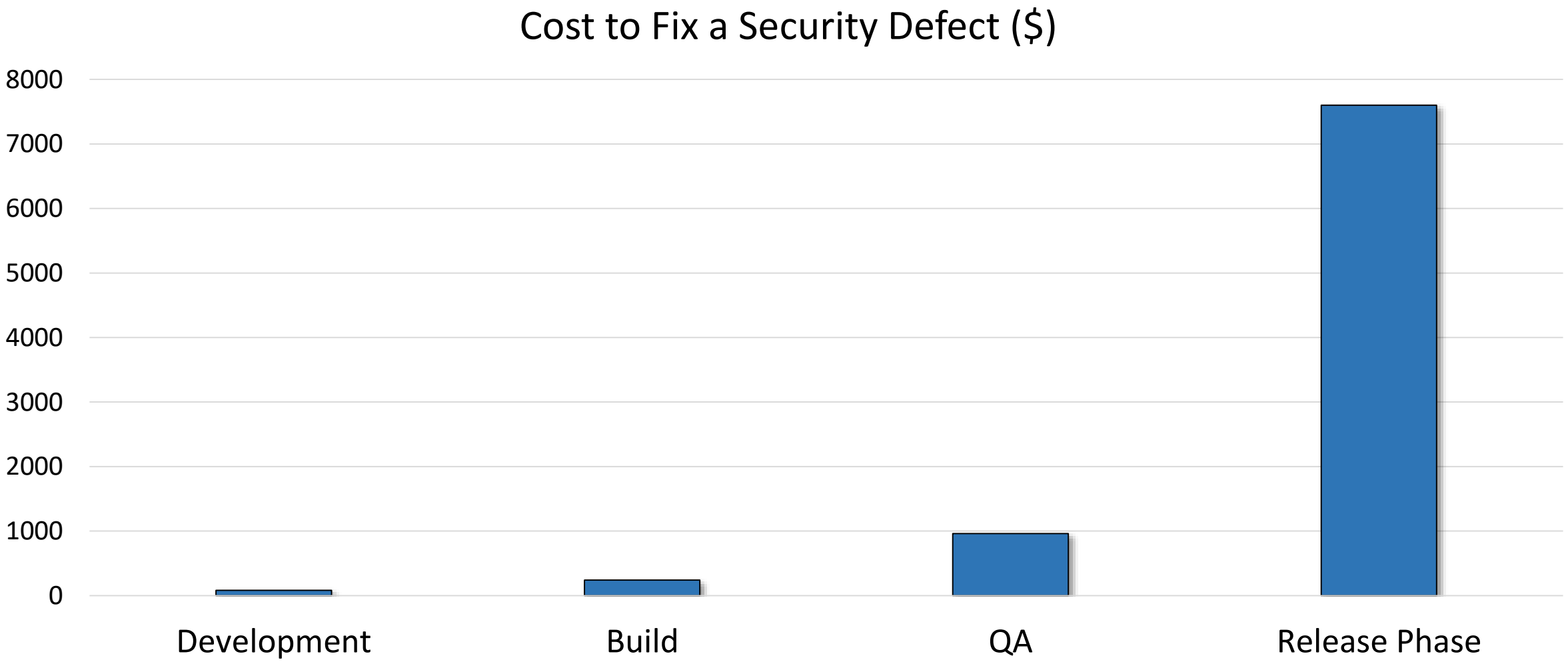
- Попробовать на синтетических примерах.
- Попробовать на маленьком проекте.
- Делать выводы, проверив стабильную версию проекта.

Статический анализ - применяй правильно!

- Подходящий инструментарий.
- Тонкая настройка.
- Инкрементальный режим.
- РЕГУЛЯРНОСТЬ.

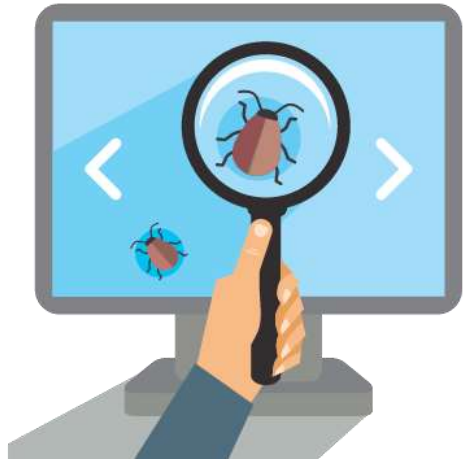


Зарелизим, прослезимся...



Типовые ошибки

- Copy-Paste
- Доступ по нулевому указателю
- Ошибочные проверки
- Выход за границу массива
- Неиспользуемый результат
- Деление на ноль
- Некомпетентность
- Сотни их…



Notepad++

```
void KeyWordsStyleDialog::updateDlg()
{
  ...
  Style & w1Style =
    _pUserLang->_styleArray.getStyler(STYLE_WORD1_INDEX);
  styleUpdate(w1Style, _pFgColour[0], _pBgColour[0],
    IDC_KEYWORD1_FONT_COMBO, IDC_KEYWORD1_FONTSIZE_COMBO,
    IDC_KEYWORD1_BOLD_CHECK, IDC_KEYWORD1_ITALIC_CHECK,
    IDC_KEYWORD1_UNDERLINE_CHECK);

  Style & w2Style =
    _pUserLang->_styleArray.getStyler(STYLE_WORD2_INDEX);
  styleUpdate(w2Style, _pFgColour[1], _pBgColour[1],
    IDC_KEYWORD2_FONT_COMBO, IDC_KEYWORD2_FONTSIZE_COMBO,
    IDC_KEYWORD2_BOLD_CHECK, IDC_KEYWORD2_ITALIC_CHECK,
    IDC_KEYWORD2_UNDERLINE_CHECK);

  Style & w3Style =
    _pUserLang->_styleArray.getStyler(STYLE_WORD3_INDEX);
  styleUpdate(w3Style, _pFgColour[2], _pBgColour[2],
    IDC_KEYWORD3_FONT_COMBO, IDC_KEYWORD3_FONTSIZE_COMBO,
    IDC_KEYWORD3_BOLD_CHECK, IDC_KEYWORD3_BOLD_CHECK,
    IDC_KEYWORD3_UNDERLINE_CHECK);

  Style & w4Style =
    _pUserLang->_styleArray.getStyler(STYLE_WORD4_INDEX);
  styleUpdate(w4Style, _pFgColour[3], _pBgColour[3],
    IDC_KEYWORD4_FONT_COMBO, IDC_KEYWORD4_FONTSIZE_COMBO,
    IDC_KEYWORD4_BOLD_CHECK, IDC_KEYWORD4_ITALIC_CHECK,
    IDC_KEYWORD4_UNDERLINE_CHECK);
  ...
}
```


```
styleUpdate(...
  IDC_KEYWORD1_BOLD_CHECK, IDC_KEYWORD1_ITALIC_CHECK,
  ...);
styleUpdate(...
  IDC_KEYWORD2_BOLD_CHECK, IDC_KEYWORD2_ITALIC_CHECK,
  ...);
styleUpdate(...
  IDC_KEYWORD3_BOLD_CHECK, IDC_KEYWORD3_BOLD_CHECK,
  ...);
styleUpdate(...
  IDC_KEYWORD4_BOLD_CHECK, IDC_KEYWORD4_ITALIC_CHECK,
  ...);
```

Chromium

```
static const int kDaysInMonth[13] = {
  0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
};

bool ValidateDateTime(const DateTime& time) {
  if (time.year < 1 || time.year > 9999 ||
      time.month < 1 || time.month > 12 ||
      time.day < 1 || time.day > 31 ||
      time.hour < 0 || time.hour > 23 ||
      time.minute < 0 || time.minute > 59 ||
      time.second < 0 || time.second > 59) {
    return false;
  }
  if (time.month == 2 && IsLeapYear(time.year)) {
    return time.month <= kDaysInMonth[time.month] + 1;
  } else {
    return time.month <= kDaysInMonth[time.month];
  }
}
```


```
if (time.month == 2 && IsLeapYear(time.year)) {
  return time.month <= kDaysInMonth[time.month] + 1;
} else {
  return time.month <= kDaysInMonth[time.month];
}
```

Mono

```
button_checked_gradient_begin = use_system_colors ? Color.Empty : Color.FromArgb (255, 223, 154);
button_checked_gradient_end = use_system_colors ? Color.Empty : Color.FromArgb (255, 166, 76);
button_checked_gradient_middle = use_system_colors ? Color.Empty : Color.FromArgb (255, 195, 116);
button_checked_highlight = Color.FromArgb (195, 211, 237);
button_checked_highlight_border = Color.FromKnownColor (KnownColor.Highlight);
button_pressed_border = use_system_colors ? Color.FromKnownColor (KnownColor.Highlight) : Color.FromArgb (0, 0, 128);
button_pressed_gradient_begin = use_system_colors ? Color.FromArgb (152, 181, 226) : Color.FromArgb (254, 128, 62);
button_pressed_gradient_end = use_system_colors ? Color.FromArgb (152, 181, 226) : Color.FromArgb (255, 223, 154);
button_pressed_gradient_middle = use_system_colors ? Color.FromArgb (152, 181, 226) : Color.FromArgb (255, 177, 109);
button_pressed_highlight = use_system_colors ? Color.FromArgb (150, 179, 225) : Color.FromArgb (150, 179, 225);
button_pressed_highlight_border = Color.FromKnownColor (KnownColor.Highlight);
button_selected_border = use_system_colors ? Color.FromKnownColor (KnownColor.Highlight) : Color.FromArgb (0, 0, 128);
button_selected_gradient_begin = use_system_colors ? Color.FromArgb (193, 210, 238) : Color.FromArgb (255, 255, 222);
button_selected_gradient_end = use_system_colors ? Color.FromArgb (193, 210, 238) : Color.FromArgb (255, 203, 136);
button_selected_gradient_middle = use_system_colors ? Color.FromArgb (193, 210, 238) : Color.FromArgb (255, 225, 172);
button_selected_highlight = use_system_colors ? Color.FromArgb (195, 211, 237) : Color.FromArgb (195, 211, 237);
button_selected_highlight_border = use_system_colors ? Color.FromKnownColor (KnownColor.Highlight) : Color.FromArgb (0, 0, 128);

check_background = use_system_colors ? Color.FromKnownColor (KnownColor.Highlight) : Color.FromArgb (255, 192, 111);
check_pressed_background = use_system_colors ? Color.FromArgb (152, 181, 226) : Color.FromArgb (254, 128, 62);
check_selected_background = use_system_colors ? Color.FromArgb (152, 181, 226) : Color.FromArgb (254, 128, 62);

grip_dark = use_system_colors ? Color.FromArgb (193, 190, 179) : Color.FromArgb (39, 65, 118);
grip_light = use_system_colors ? SystemColors.Window : Color.FromArgb (255, 255, 255);

image_margin_gradient_begin = use_system_colors ? Color.FromArgb (251, 250, 246) : Color.FromArgb (227, 239, 255);
image_margin_gradient_end = use_system_colors ? SystemColors.Control : Color.FromArgb (123, 164, 224);
image_margin_gradient_middle = use_system_colors ? Color.FromArgb (246, 244, 236) : Color.FromArgb (203, 225, 252);
image_margin_revealed_gradient_begin = use_system_colors ? Color.FromArgb (247, 246, 239) : Color.FromArgb (203, 221, 246);
image_margin_revealed_gradient_end = use_system_colors ? Color.FromArgb (238, 235, 220) : Color.FromArgb (114, 155, 215);
image_margin_revealed_gradient_middle = use_system_colors ? Color.FromArgb (242, 240, 228) : Color.FromArgb (161, 197, 249);
```

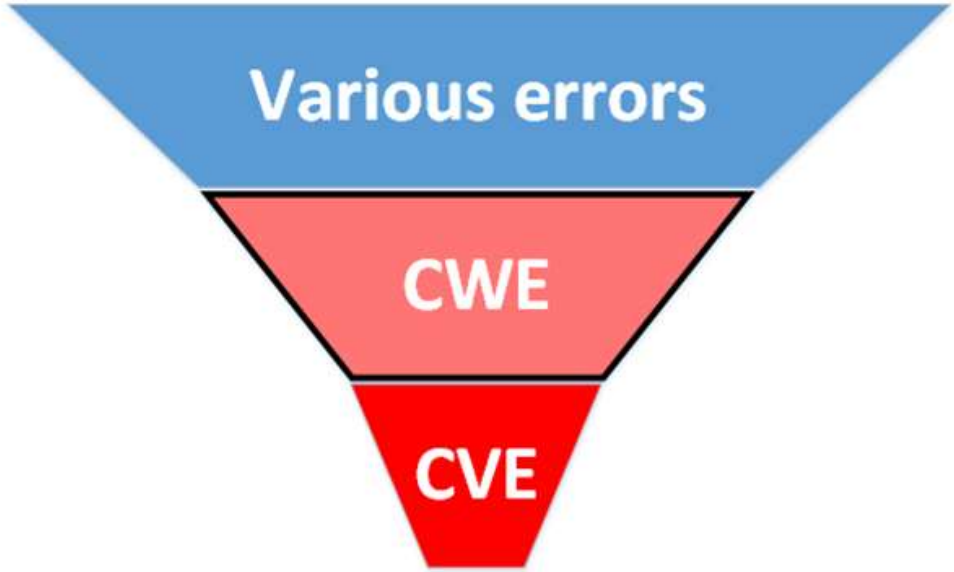
problem???

```
button_pressed_highlight = use_system_colors ?
                           Color.FromArgb (150, 179, 225) :
                           Color.FromArgb (150, 179, 225);
```

А что с уязвимостями?

CWE - Common Weakness Enumeration

CVE - Common Vulnerabilities and Exposures



CVE-2014-1266


```
static OSStatus
SSLVerifySignedServerKeyExchange(.....)
{
  OSStatus err;
  ....

  if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
  if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
  if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
  ....

fail:
  SSLFreeBuffer(&signedHashes);
  SSLFreeBuffer(&hashCtx);
  return err;
}
```

CVE-2014-1266:
The SSLVerifySignedServerKeyExchange function in libsecurity_ssl/lib/sslKeyExchange.c in the Secure Transport feature in the Data Security component in Apple iOS 6.x ..... does not check the signature in a TLS Server Key Exchange message, which allows man-in-the-middle attackers to spoof SSL servers by (1) using an arbitrary private key for the signing step or (2) omitting the signing step.

PVS-Studio сообщает сразу о двух аномалиях:

- V640 The code's operational logic does not correspond with its formatting. The statement is indented to the right, but it is always executed. It is possible that curly brackets are missing.
- V779 Unreachable code detected. It is possible that an error is present

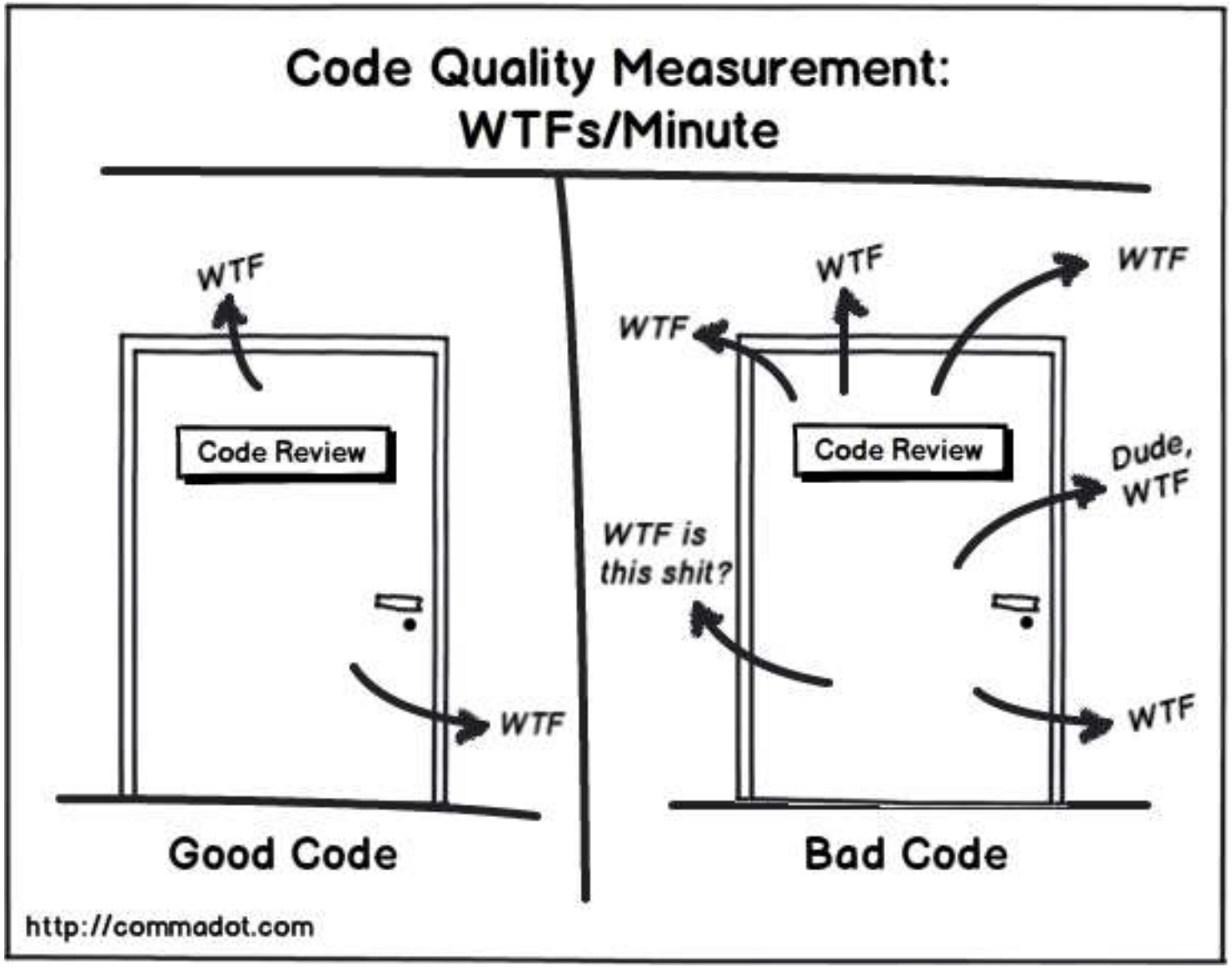
Ограничения SAST

- Слабое диагностирование утечек памяти и параллельных ошибок.
- Высокий процент ложных срабатываний.

Мифы, связанные с SAST

- Дорого.
- Не для новичков.
- Сложно внедрять на большом проекте.
- Панацея от всех бед.

Одна из альтернатив



Альтернативы

Административные:

- Делать сразу правильно (не работает).
- Следование корпоративным правилам.
- Использование «лучших практик».
- Парная разработка.
- Разработка через тестирование (TDD).
- Гибкая разработка Agile.

Альтернативы

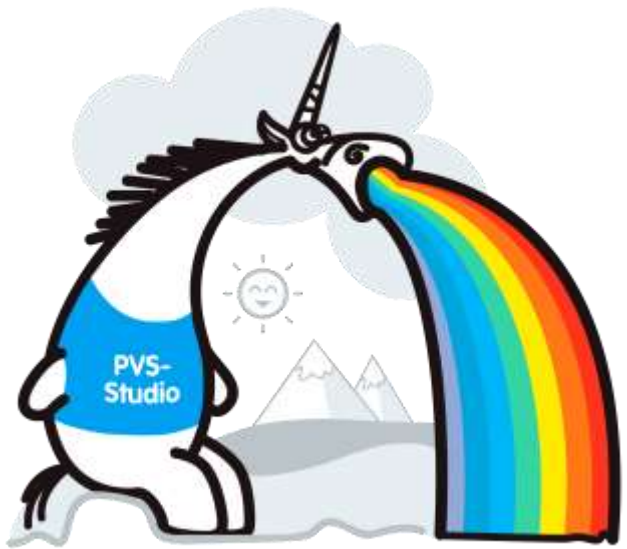
Инструментальные средства:

- Нагрузочные тесты.
- Unit-тесты.
- Динамические анализаторы.

Выводы

- Не все опечатки станут уязвимостями.
- Многие уязвимости являются опечатками.
- SAST не идеален, но это не повод его не использовать.
- Используйте инструменты SAST правильно.
- Своевременность: не допустите перехода затрат в расплату.
- Не ищите «серебряной пули».

Ответы на вопросы



Контакты

Сергей Хренов

C# разработчик, PVS-Studio

hrenov@viva64.com

www.viva64.com