Андрей Карпов

Лекция 7/12

Статический анализ как неотъемлемая часть разработки при написании программ на C++

Докладчик

Карпов
Андрей Николаевич

- Сооснователь компании PVS-Studio, технический директор
- Разработал первые реализации ядра C++ анализатора
- Автор статей о качестве кода и статическом анализе

Зачем это нужно

- Программы становятся больше и сложнее
- Старые подходы к обеспечению качества недостаточны
- Статический анализ, как один из способов повышения качества кода
- Почему это важно и неизбежно
- Статический анализ как часть DevOps
- Хороший программист использует статический анализ кода

Анекдот

Жила-была девочка, и была у нее кофточка. И когда девочка надевала кофточку, кофточка начинала душить девочку. Пошла девочка к маме и рассказала ей обо всем. Мама выслушала ее и купила ей другую кофточку.

И в самом деле, нельзя же одну кофточку носить десять лет.

Размер проектов растёт

Дискетки с архивами → Система контроля версий

«Я ведь просил тебя исправить» → Багтрекер

Ручная сборка → Continuous Integration

Обзоры кода это хорошо, но недостаточно

- MS DOS 1.0 : **4 000** строк кода
- Ядро Linux 1.0.0 : **176 250** строк кода
- Ядро Linux 5.0 в **150** раз больше: Более **26 000 000** строк кода

- Photoshop 1.0 : **128 000** строк кода
- Photoshop CS 6 : **10 000 000** строк кода

Взаимосвязи в большом проекте

```
PUGI__FN 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);
}
```

Анализатор PVS-Studio говорит, что используется неинициализированный буфер buf.
Ложное срабатывание?

Подробно: https://www.viva64.com/ru/b/0535/

Взаимосвязи в большом проекте

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

```
PUGI__FN 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);
}
```

Плотность ошибок

Размер проекта (число строк кода)	Типичная плотность ошибок
Менее 2K	**0 – 25** ошибок на 1000 строк кода
2K – 16K	**0 – 40** ошибок на 1000 строк кода
16K – 64K	**0.5 – 50** ошибок на 1000 строк кода
64K – 512K	**2 – 70** ошибок на 1000 строк кода
512K и более	**4 - 100** ошибок на 1000 строк кода

"Program Quality and Programmer Productivity" (Jones, 1977),
"Estimating Software Costs" (Jones, 1998).

Статический анализ кода

Статический анализ кода

- Не заменяет, но дополняет обзоры кода
- Позволяет совладать с качеством кода в большом проекте
- Использование
  - Поиск ошибок/потенциальных уязвимостей
  - Обучение
  - Оценка проекта

PVS-Studio

PVS-Studio - это инструмент для выявления ошибок и потенциальных уязвимостей в исходном коде программ, написанных на языках C, C++, C# и Java.

Работает в 64-битных системах на Windows, Linux и macOS и может анализировать код, предназначенный для 32-битных, 64-битных и встраиваемых ARM платформ.

Статический анализ: поиск ошибок

- Вместо многих слов сразу рассмотрим интересный пример
- Подробности смотрите в статье “31 февраля” - https://www.viva64.com/ru/b/0550/

```
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];
  }
}
```

Protocol Buffers
(protobuf)

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

- V547 / CWE-571 Expression 'time.month <= kDaysInMonth[time.month] + 1' is always true. time.cc 83
- V547 / CWE-571 Expression 'time.month <= kDaysInMonth[time.month]' is always true. time.cc 85

```
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];
  }
}
```

Как работают анализаторы кода

- Сопоставление с шаблоном
- Анализ потока данных
- Символьное выполнение
- Аннотирование методов
- Автоматическое аннотирование методов

Как работают анализаторы кода

- Технологии, используемые в анализаторе кода PVS-Studio для поиска ошибок и потенциальных уязвимостей
  https://www.viva64.com/ru/b/0592/

- Yappi Days. "Опыт разработки статического анализатора кода"
  https://youtu.be/tRzxDFZhy1w

Примеры обнаруживаемых ошибок

Android, C

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

Где ошибка?

Android, C

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

PVS-Studio: V597 CWE-14 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

Компилятор удаляет код затирания буфера

- При отладке Debug-версии этой ошибки нет
- Если не знаешь про этот паттерн, то нельзя найти ошибку во время обзора кода
- Невозможно написать тест, если не искать эту ошибку целенаправленно
- Эта ошибка везде. Я находил её в таких проектах, как: Android, XNU kernel, MySQL, Sphinx, Tizen, FreeBSD Kernel, Linux Kernel, Haiku Operation System, Qt, Apache HTTP Server и т.д.

```
int
UserlandFS::KernelEmu::new_path(const char *path, char **copy)
{
  ....
  // append a dot, if desired
  if (appendDot) {
    copiedPath[len] = '.';
    copiedPath[len] = '\0';
  }
  ....
}
```

PVS-Studio: V519 The 'copiedPath[len]' variable is assigned values twice successively. Perhaps this is a mistake. Check lines: 92, 93. kernel_emu.cpp 93

```
value.start_fragment_union_rect.size.width =
  std::max(descendant.offset_to_container_box.left +
        descendant.fragment->Size().width -
        value.start_fragment_union_rect.offset.left,
      value.start_fragment_union_rect.size.width);
value.start_fragment_union_rect.size.height =
  std::max(descendant.offset_to_container_box.top +
        descendant.fragment->Size().height -
        value.start_fragment_union_rect.offset.top,
      value.start_fragment_union_rect.size.width);
```

```
value.start_fragment_union_rect.size.width =
  std::max(descendant.offset_to_container_box.left +
        descendant.fragment->Size().width -
        value.start_fragment_union_rect.offset.left,
      value.start_fragment_union_rect.size.width);
value.start_fragment_union_rect.size.height =
  std::max(descendant.offset_to_container_box.top +
        descendant.fragment->Size().height -
        value.start_fragment_union_rect.offset.top,
      value.start_fragment_union_rect.size.width);
```

PVS-Studio: V778 CWE-682 Two similar code fragments were found. Perhaps, this is a typo and 'height' variable should be used instead of 'width'. ng_fragment_builder.cc 326

FreeBSD Kernel, C

```
#define  AE_IDLE_TIMEOUT     100
....
int i;
....
/* Wait for IDLE state. */
for (i = 0; i < AE_IDLE_TIMEOUT; i--) {
  val = AE_READ_4(sc, AE_IDLE_REG);
  if ((val & (AE_IDLE_RXMAC | AE_IDLE_DMAWRITE)) == 0)
    break;
  DELAY(100);
}
```

PVS-Studio: V621 Consider inspecting the 'for' operator. It's possible that the loop will be executed incorrectly or won't be executed at all. if_ae.c 1663

```
Init *TGParser::ParseValue(Record *CurRec, RecTy *ItemType,
                           IDParseMode Mode) {
  ....
  TypedInit *LHS = dyn_cast<TypedInit>(Result);
  ....
  LHS = dyn_cast<TypedInit>(
    UnOpInit::get(UnOpInit::CAST, LHS, StringRecTy::get())
      ->Fold(CurRec));
  if (!LHS) {
    Error(PasteLoc, Twine("can't cast '") + LHS->getAsString() +
                    "' to string");
    return nullptr;
  }
  ....
}
```

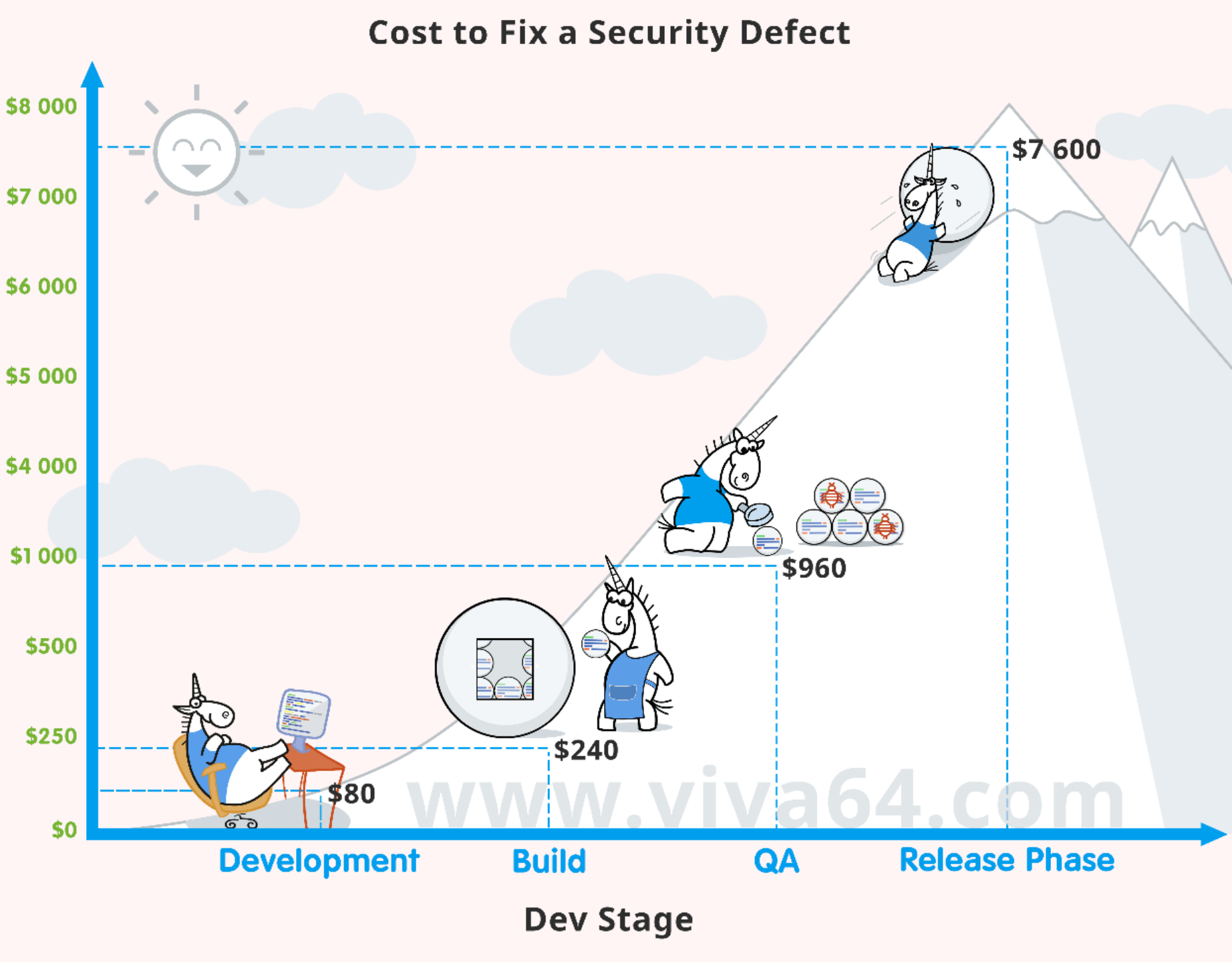
LLVM

PVS-Studio: V522 [CWE-476] Dereferencing of the null pointer 'LHS' might take place. TGParser.cpp 2152

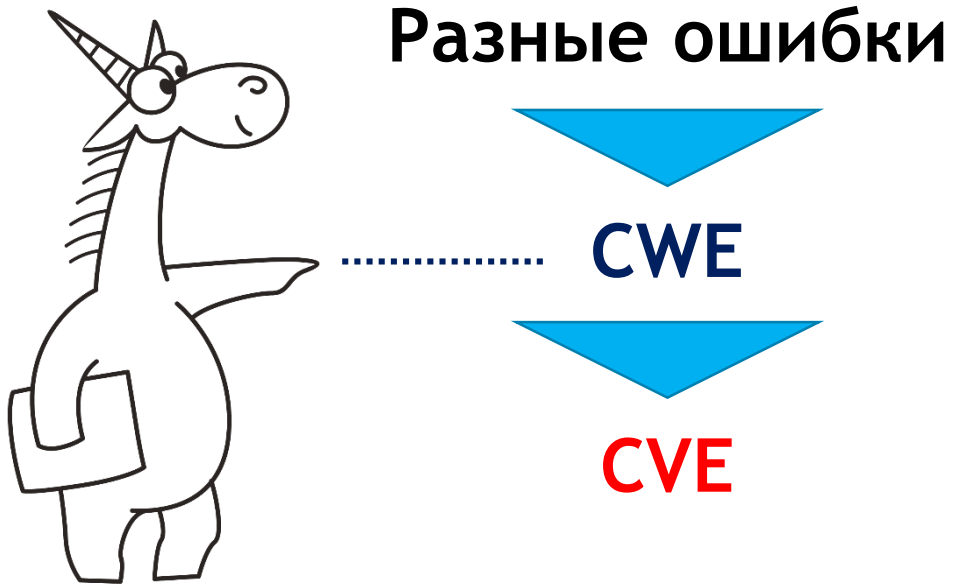
Неправильное/правильное использование

- Изучение: синтетические тесты / реальные проекты
- Включить всё / постепенно настраивать (массовое подавление)
- Разовые проверки / регулярное использование

SAST


Cost to Fix a Security Defect
$8 000
$7 000
$6 000
$5 000
$4 000
$1 000
$500
$250
$0
$7 600
$960
$240
$80
Development
Build
QA
Release Phase
Dev Stage
www.viva64.com

Предотвращение уязвимостей




```
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

Предотвращение уязвимостей

PVS-Studio сообщает сразу о двух аномалиях

- V640 / CWE-483 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 / CWE-561 Unreachable code detected. It is possible that an error is present

Подробнее:
Как PVS-Studio может помочь в поиске уязвимостей?
https://www.viva64.com/ru/b/0514/

Выводы

- Статический анализ кода – это добро
- У этой методологии, как и у любой другой, есть недостатки
- Однако это не повод её не использовать
- Аналогия: проверка орфографии и грамматики в Microsoft Office

Дополнительно

- Дискуссия о статическом анализе кода
  https://www.viva64.com/ru/b/0545/
- Философия статического анализа кода: у нас 100 программистов, анализатор нашел мало ошибок, он бесполезен?
  https://www.viva64.com/ru/b/0534/
- Статья о статическом анализе кода для менеджеров, которую не стоит читать программистам
  https://www.viva64.com/ru/b/0498/
- Ошибки, которые не находит статический анализ кода, потому, что он не используется
  https://www.viva64.com/ru/b/0639/

END

Q&A