Как работает анализ потока данных в статическом анализаторе кода

Павел Беликов
Разработкик PVS-Studio C++
belikov@viva64.com

PVS-Studio

- Статический анализатор C, C++, C# кода
- Работает на Windows, Linux, macOS
- Плагин для Visual Studio
- Интегрируется в SonarQube и Jenkins
- Быстрый старт (Standalone, pvs-studio-analyzer)

Содержание

- Основные виды и задачи Data Flow анализа
- Анализ условий
- Анализ циклов
- Символьное выполнение
- Примеры ошибок, найденных в реальных проектах

Что такое data flow analysis

- Вычисляем множество значений выражения или его свойства
  - Числа
  - Нулевой/ненулевой указатель
  - Строки
  - Размер и содержимое контейнеров/optional
- Определяем состояние переменных

Основные задачи

- Множество значений должно быть надмножеством реального
- Время ограничено
- Количество ложных срабатываний должно быть минимизировано

А зачем оно нужно?

```
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];
  }
}
```

А зачем оно нужно?

```
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];
  }
}
```

Protobuf

- 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

Основное уравнение

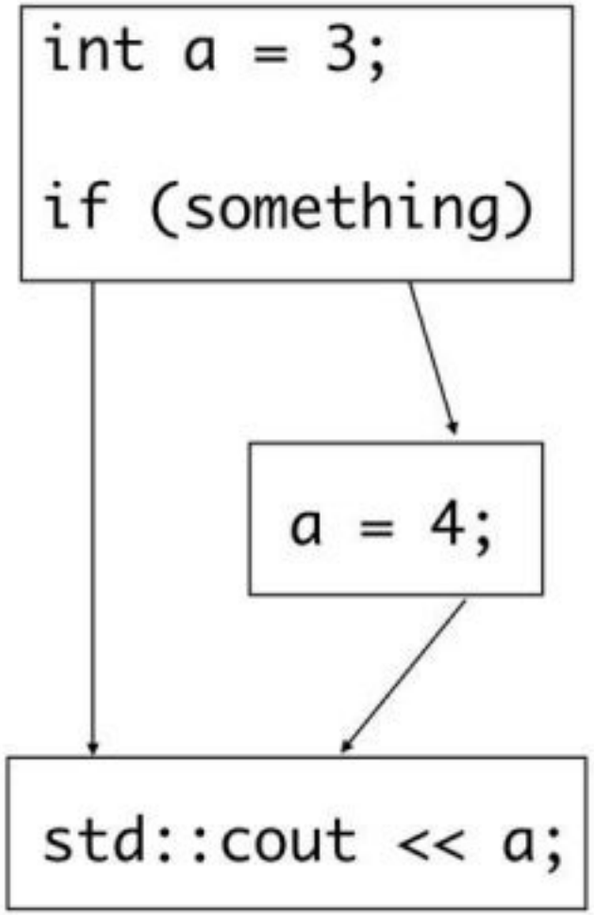
$$out_b = trans_b(in_b)$$
$$in_b = join_{p \in pred_b}(out_p)$$

- b - блок кода
- in/out - состояния переменных на входе и выходе из блока
- trans - функция меняющая состояние переменных в блоке
- join - функция, объединяющая состояние переменных в разных путях выполнения

Пример

```
int a = 3;

if (something)

{

        a = 4;

}

std::cout << a;
```

Пример



in = {}, out = {a=3}

in = {a=3}, out = {a=4}

in = {a=3}∪{a=4}={a=[3;4]}

Flow sensitivity

- Flow-sensitive анализ зависит от порядка выражений в коде
- Пример flow-insensitive анализа: поиск модифицированных переменных в блоке
- Нужен способ обхода кода

Flow sensitivity

- Data Flow работает с Control Flow Graph
- На практике можно использовать AST
- AST проще и понятнее большинству программистов
- Для AST больше инструментов, парсеры выдают AST
- CFG можно моделировать поверх AST

Flow sensitivity

- Forward analysis
  - Передаём информацию блоку B от предшествующих ему блоков
  - Хорошо подходит для вычисления переменных и определения reaching definitions
- Backward analysis
  - Передаём информацию от блока B предшествующим ему блокам
  - Хорошо подходит для live variable analysis

Пример backward analysis

```
__private_extern__ void
YSHA1Transform(u_int32_t state[5],
               const unsigned char buffer[64])
{
  u_int32_t a, b, c, d, e;
  ....
  state[0] += a;
  state[1] += b;
  state[2] += c;
  state[3] += d;
  state[4] += e;
  /* Wipe variables */
  a = b = c = d = e = 0;
}
```

XNU kernel

V1001 CWE-563 The 'a' variable is assigned but is not used until the end of the function. sha1mod.c 120

Пример forward analysis

- Reachable definitions

$$\mathrm{REACH}_{\mathrm{in}}[S] = \bigcup_{p \in pred[S]} \mathrm{REACH}_{\mathrm{out}}[p]$$

$$\mathrm{REACH}_{\mathrm{out}}[S] = \mathrm{GEN}[S] \cup (\mathrm{REACH}_{\mathrm{in}}[S] - \mathrm{KILL}[S])$$

- REACH - множество определений переменной, которые могут прочитаны в выражении S
- GEN - новые определения
- KILL - “убитые” определения

Пример forward analysis

```
ParseResult ParseOption (string option, ref string[] args , CompilerSettings settings) {
    AssemblyResource res = null;                                              GEN={res0}
    switch (s.Length) {
        case 1:
            res = new AssemblyResource (s[0], Path.GetFileName (s[0]));       GEN={res1}, KILL={res0}
            break;
        case 2:
            res = new AssemblyResource (s[0], s[1]);                          GEN={res2}, KILL={res0}
            break;
        default:
            report.Error (-2005, "Wrong number of arguments for option '{0}'", option);
            return ParseResult.Error;
    }
    if (res != null) { ... }                                                  REACH={res1, res2}
}
```

ILSpy

V3022 Expression 'res != null' is always true. settings.cs 827

Must vs may

- Must
  - Data flow факт должен быть верным для всех путей
  - Выражается через пересечение множеств
- May
  - Факт должен быть верен хотя бы для одного пути
  - Выражается через объединение множеств

Must vs may

- Статический анализ часто работает с may
- Никто не пишет

```
int *p = nullptr;

if (something) p = nullptr;

else if (something_else) p = nullptr;

else p = nullptr;

*p = 42;
```

Must vs may

```
STDMETHODIMP sdnAccessible::get_computedStyle(
                    BSTR __RPC_FAR* aStyleProperties,
                    BSTR __RPC_FAR* aStyleValues,
                    unsigned short __RPC_FAR* aNumStyleProperties)
{
  if (!aStyleProperties || aStyleValues || !aNumStyleProperties)
    return E_INVALIDARG;
  ....
  aStyleValues[realIndex] = ::SysAllocString(value.get());
  ....
}
```

Mozilla Thunderbird

V522 Dereferencing of the null pointer ‘aStyleValues’ might take place. sdnaccessible.cpp 252

Path-sensitive analysis

- May на одном из путей мало
- Что если путь невозможен?
- Нужно анализировать условия!

Path-sensitive analysis

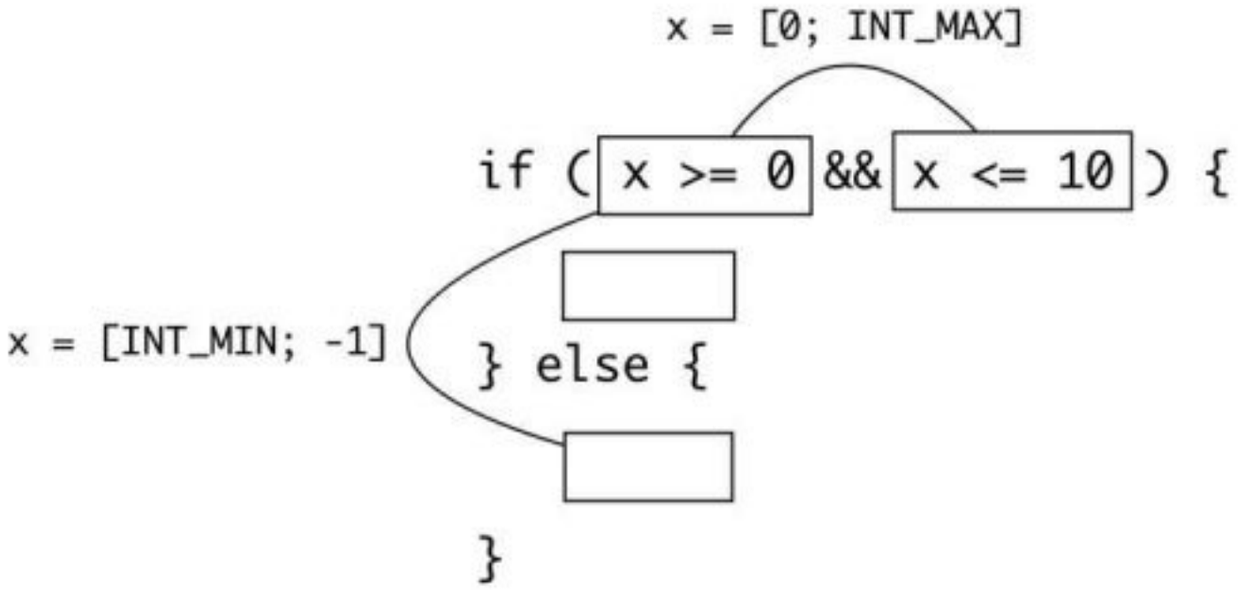
```
enum {
  Runesync  = 0x80,
  Runeself  = 0x80,
};

char* utfrune(const char *s, int c) {
  ....
  if (c < Runesync) return strchr(s, c);  // c: then [INT_MIN; 0x79] else [0x80; INT_MAX]
  for(;;) {
    c1 = *(unsigned char*)s;
    if (c1 < Runeself) {                  // c1: then [0; 0x79]
      if (c1 == 0) return 0;              // c1: then 0 else [1; 0x79]
      if (c1 == c) return (char*)s;       // if ([1; 0x79] == [0x80; INT_MAX])
      ....
    }
    ....
  }
  return 0;
}
```

RE2 V547 CWE-570 Expression 'c1 == c' is always false. rune.cc 247

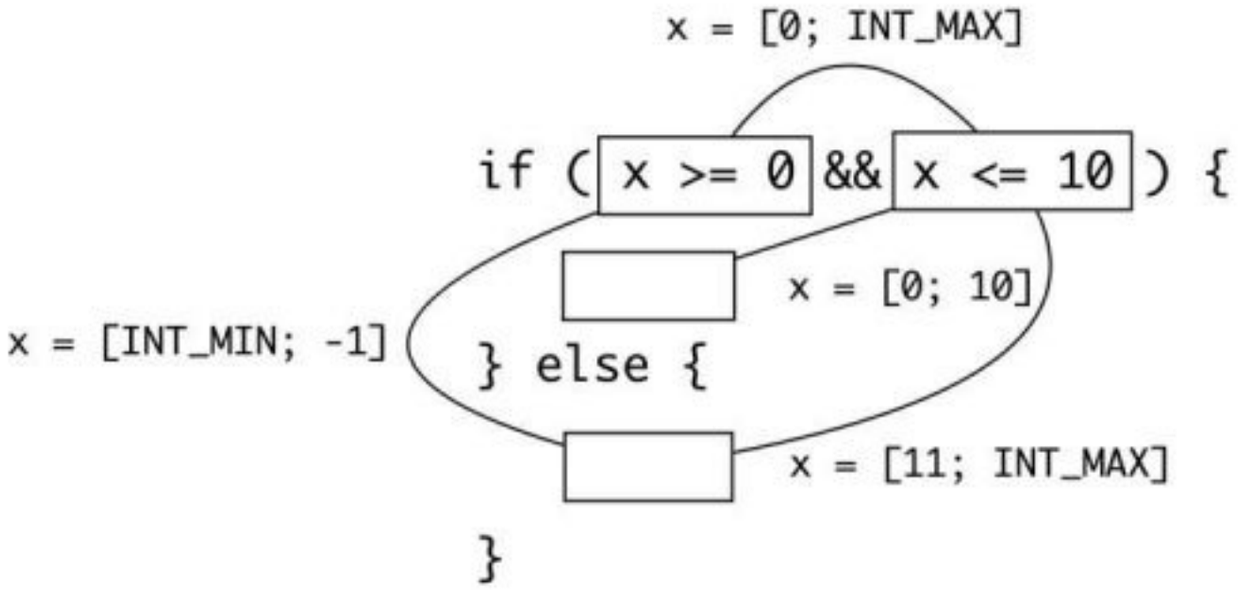
Short circuit

```
if ( x >= 0 && x <= 10 ) {
    
} else {
    
}
```

Short circuit



Short circuit



then: x = [0; 10]
else: x = [INT_MIN; -1] ∪ [11; INT_MAX]

Short circuit

```
internal bool SafeForExport()
{
  return DisplayEntry.SafeForExport() &&
         ItemSelectionCondition == null
      || ItemSelectionCondition.SafeForExport();
}
```

PowerShell

V3080 Possible null dereference. Consider inspecting 'ItemSelectionCondition'. System.Management.Automation displayDescriptionData_List.cs 352

Join problem

```
int *p;
if (condition) {
  p = new int;
} else {
  p = nullptr;
}

// p - nullable

if (condition) {
  *p = 42; // null dereference?
}
```

Join problem

- При объединении путей мы теряем информацию
- Лучше откладывать объединение состояний как можно дольше
- Но есть проблема с path explosion

Join problem

```
int *p;
if (condition) {
  p = new int;  // p = non null if condition
} else {
  p = nullptr;  // p = null if !condition
}

// p = non null if condition
//   ∪ null     if !condition

if (condition) {
  // p = non null
  *p = 42;
}
```

Join problem

```
int a, b;
if (condition) {
  a = 1;
  b = 2;
} else {
  a = 2;
  b = 1;
}

return a + b; // a = 1 if condition ∪ 2 if !condition
              // b = 2 if condition ∪ 1 if !condition
              // a + b = 3 if condition ∪ 3 if !condition
              // a + b = 3
```

Try-catch

```
try {
  SomeClass c(someFunction(), 42);
  c.foo();
  return c + “abc”;
} catch (...) {
}
```

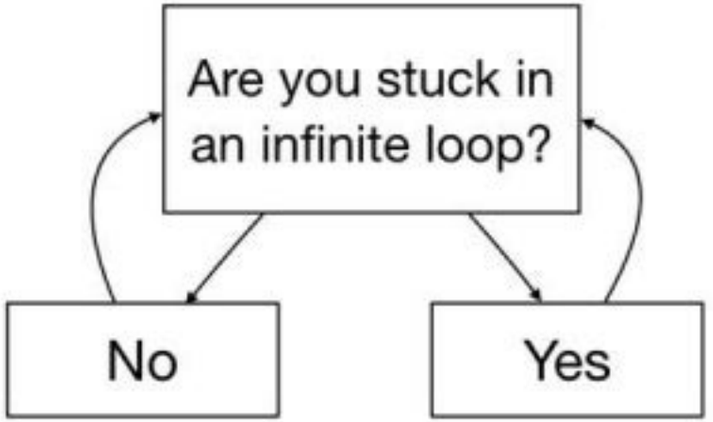
Try-catch

```
try {
  SomeClass c(someFunction(), 42);
  c.foo();
  return c + “abc”;
} catch (...) {
}
```

- вызов someFunction()
- конструктор c
- вызов метода foo()
- конструкторы временных объектов
- operator +
- конструктор для возвращаемого объекта
- деструкторы для временных объектов
- деструктор c

Loop analysis

- В общем случае анализировать сложно и медленно
- Анализируем первую итерацию отдельно
- “Убиваем” все новые определения переменных после цикла



Loop invariants

```
public final R getSomeBuildWithWorkspace() {
  int cnt=0; // <= определение переменной вне цикла
  for (R b = getLastBuild(); cnt<5 && b!=null; b=b.getPreviousBuild()) {
    FilePath ws = b.getWorkspace();
    if (ws!=null) return b;
  }
  return null;
}
```

Jenkins
V6022 Expression 'cnt < 5' is always true AbstractProject.java 557

Первая итерация

```
void Measure::read(XmlReader& e, int staffIdx) {
  Segment* segment = 0;
  ....
  while (e.readNextStartElement()) {
    const QStringRef& tag(e.name());

    if (tag == "move")
      e.initTick(e.readFraction().ticks() + tick());
    ....
    else if (tag == "sysInitBarLineType") {
      ....
      segment = getSegmentR(SegmentType::BeginBarLine, 0); // !!!
      segment->add(barLine);                               // <= OK
    }
    ....
    else if (tag == "Segment")
      segment->read(e);                                    // <= ERROR
    ....
  }
}
```

MuseScore V522 Dereferencing of the null pointer 'segment' might take place. measure.cpp 2220

Loop control flow

```
SkOpSpan* SkOpContour::undoneSpan() {
  SkOpSegment* testSegment = &fHead;
  bool allDone = true;
  do {
    if (testSegment->done()) {
      continue;
    }
    allDone = false;
    return testSegment->undoneSpan();
  } while ((testSegment = testSegment->next()));
  if (allDone) {
    fDone = true;
  }
  return nullptr;
}
```

Skia Graphics Engine
V547 CWE-571 Expression 'allDone' is always true. skopcontour.cpp 43

Loop control flow

```
SkOpSpan* SkOpContour::undoneSpan() {
  SkOpSegment* testSegment = &fHead;
  bool allDone = true;
  do {
    if (testSegment->done()) {
      continue;
    }
    allDone = false; // <= этот путь не учитываем
    return testSegment->undoneSpan();
  } while ((testSegment = testSegment->next()));
  if (allDone) {
    fDone = true;
  }
  return nullptr;
}
```

Skia Graphics Engine
V547 CWE-571 Expression 'allDone' is always true. skopcontour.cpp 43

Loop control flow

```
int *p = nullptr;
for (int *p2 : array) {
  if (*p2 % 42 == 0) {
    p = p2;
    break;
  }
}

*p *= *p;
```

Loop control flow

```
int *p = nullptr;
for (int *p2 : array) { // p == nullptr
  if (*p2 % 42 == 0) {
    p = p2;
    break; // p = not null
  }
}

*p *= *p; // p - nullable
```

Loop counter analysis

```
for (int i = 0; i < 10; ++i)
{
    // i = [INT_MIN; 9] ?
    // i = [0; 9] !!!
}
```

Loop counter analysis

```
#define  AE_IDLE_TIMEOUT    100

static void
ae_stop_rxmac(ae_softc_t *sc)
{
  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);
  }
  ....
}
```

FreeBSD Kernel

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

Есть проблема

```
for (int i = 0; i < n; ++i) {
  for (int j = i + 1; j < n; ++j) {
   // j - i
  }
}
```

Есть проблема

```
int i = /* [0; 42] */;
int j = i + 1; // [1; 43]
int r = j - i; // [-43; 41]???
```

Symbolic execution

```
int i = /* [0; 42] */;
int j = i + 1; // [1; 43]
int r = j - i; // i + 1 - i = 1
```

Symbolic execution

- Вычисляем всё в символьных выражениях
- Составляем систему уравнений
- Загоняем её в SMT-solver
- ???
- PROFIT

Symbolic execution

```
public static MMMethodKind valueOf(....) {
  MMMethodKind result = OTHER;
  for (MMMethodKind k : values()) {
    if (k.detector.test(method) && result.level < k.level) {
      if (result.level == k.level) {
        throw new SpoonException(....);
      }
      result = k;
    }
  }
  return result;
}
```

Spoon
MMMethodKind.java:129: V6007 Expression 'result.level == k.level' is always false.

Оптимизаторы vs статические анализаторы

- Цель оптимизатора - сохранить корректность кода при любых входных данных
- Статический анализатор может “сжульничать” для того, чтобы находить больше ошибок
- В худшем случае, найдём code smell

UB

```
static char *
read_string (FILE *fp)
{
  char buf[MAX_STRING_LEN];

  if (!fgets (buf, MAX_STRING_LEN, fp)) {
    return 0;
  }

  if (strlen (buf) < MAX_STRING_LEN) {
    if (strlen (buf)) {
      buf[strlen (buf)-1] = 0;
    }
    return strdup (buf);
  } else {
    return 0;
  }
}
```

Ardour

V547 Expression 'strlen(buf) < 256' is always true. vst_info_file.cc 262

UB

```
void HttpAuthHandlerRegistryFactory::RegisterSchemeFactory(
    const std::string& scheme,
    HttpAuthHandlerFactory* factory)
{
  factory->set_http_auth_preferences(http_auth_preferences());
  std::string lower_scheme = base::ToLowerASCII(scheme);
  if (factory)
    factory_map_[lower_scheme] = base::WrapUnique(factory);
  else
    factory_map_.erase(lower_scheme);
}
```

Chromium
V595 CWE-476 The 'factory' pointer was utilized before it was verified against nullptr. Check lines: 122, 124. http_auth_handler_factory.cc 122

Pointer analysis

- *p = 42;
- Можем сбрасывать всю накопленную информацию
- Может помочь pointer analysis

Pointer analysis

```
int a = 0;
int b = 0;
int &ref = condition ? a : b;
ref = 42; // <= ref указывает на a или b
// a = U, b = U
```

Pointer analysis

```
void foo(int &, int &);
int &bar();

int a = 0;
int b = 0;
foo(a, b);
a = 42;
b = 42;
int &ref = bar();
ref = 0; // <= ref может указывать на a или b.
```

Context sensitive

- foo();
- Можем сбрасывать всю накопленную информацию
- Популярные библиотеки аннотируем
- Радуемся 10 способам передать переменную в функцию
- Продолжаем жульничать и эксплуатировать UB

Context sensitive

- анализ функции с учётом контекста вызывающей стороны
- плохо масштабируется
- полезен для анализа небольших функций (getters/setters, например)

Context sensitive

```
void foo(int *p) { // анализируем два раза
  *p = 42;
}

void bar() {
 int *p = something ? new int : nullptr;
 foo(p); // повторно анализируем foo и находим ошибку
}
```

Context insensitive

```
void foo(int *p) { // p != nullptr
  *p = 42;
}

void bar() {
 int *p = something ? new int : nullptr;
 foo(p); // нарушен контракт p != nullptr, нашли ошибку
}
```

Context insensitive

```
int foo(int *p, bool c, int a, int b) {
  int res;
  if (c) {
    *p = 42;
   res = a + b;
  } else {
   res = a - b;
  }
  return res;
}

// expects:
//   p - not null, if c
// returns:
//   a + b, if c
//   a - b, if !c
```

Context insensitive

- Анализируем тело функции, составляем её аннотацию
  - Контракт на аргументы
  - Наличие глобального состояния
  - Возвращаемое значение
  - И многое другое
- contracts proposal

```
void foo(const std::vector<int> &indices)
[[expects: !indices.empty()]];
```

Context insensitive

```
static int netagent_send_error_response(struct netagent_session *session, ....) {
  int error = 0;
  u_int8_t *response = NULL;
  size_t response_size = sizeof(struct netagent_message_header);
  MALLOC(response, u_int8_t *, response_size, M_NETAGENT, M_WAITOK);
  if (response == NULL) return (ENOMEM);
  (void)netagent_buffer_write_message_header(.....);
  if ((error = netagent_send_ctl_data(session->control_unit, (u_int8_t *)response, response_size)))
  ....
 }

static void netagent_handle_unregister_message(struct netagent_session *session, ....)
  u_int32_t response_error = NETAGENT_MESSAGE_ERROR_INTERNAL;
  if (session == NULL) {
    NETAGENTLOG0(LOG_ERR, "Failed to find session");
    response_error = NETAGENT_MESSAGE_ERROR_INTERNAL;
    goto fail;
  }

  ....
  return;
fail:
  netagent_send_error_response(session, NETAGENT_MESSAGE_TYPE_UNREGISTER, message_id, response_error);
}
```

XNU kernel

V522 CWE-628 Dereferencing of the null pointer 'session' might take place. The null pointer is passed into 'netagent_send_error_response' function. Inspect the first argument. Check lines: 427, 972. network_agent.c 427

Итоги

- Data flow analysis - полезная методика нахождения ошибок
- Для поиска ошибок приходится оперировать большим и местами странным множеством свойств
- Сочетание разных техник позволяет повысить достоверность результатов анализа
- Различные эвристики и предположения позволяют находить больше ошибок
- Любой серьёзный статический анализатор должен использовать data flow analysis

Ответы на вопросы

PVS-Studio: http://www.viva64.com/ru/pvs-studio/