
PVS-Studio

Go-Go-Gadg...Error? Смотрим, как ошибаются Go разработчики!

PVS-Studio

Глеб Асламов

Senior Developer Advocate

Статический анализ в мире go

Статический анализ в Go

Статический анализ в Go

- Статический анализ это процесс выявления недочетов, ошибок и потенциальных уязвимостей в исходном коде программ. Статический анализ можно рассматривать как автоматизированный процесс обзора кода.

Статический анализ в Go

- Статический анализ это процесс выявления недочетов, ошибок и потенциальных уязвимостей в исходном коде программ. Статический анализ можно рассматривать как автоматизированный процесс обзора кода.
- Но я думаю вы это и лучше меня знаете...
  Ведь так?

Немного базовой-базы

- go vet *как* часть тулчейна

Немного базовой-базы

- go vet ***это*** часть тулчейна

  Нет, не вашего
  Самого тулчейна Go, это не отдельный линтер

Немного базовой-базы

- go vet ***это*** часть тулчейна

  Нет, не вашего
  Самого тулчейна Go, это не отдельный линтер

- Был go build? go test? Вы уверенный пользователь статического анализатора!

Вы его уже используете

Что мы будем сегодня рассматривать

1. Откуда мы это взяли
2. Ошибаемся как все
3. А зачем это все, если есть go vet
4. Ошибаемся как go'шник
5. И что с этим делать?

Откуда мы это взяли?

Мы тут недавно начали...

#Go 29 Дек 2025

Как сделать свой статический анализатор для Go?

Артём Ровенский

Go разработчики постоянно сталкиваются с предупреждениями встроенного статического анализатора. А что делать, если его возможностей не хватает или нужно искать что-то специфичное для вашего...

...

#Go 11 Мар 2026

Бета-тест Go анализатора PVS-Studio

Артём Ровенский

Команда PVS-Studio усердно работала над созданием статического анализатора кода для Go, и наконец мы рады объявить о начале открытого тестирования! Приглашаем всех заинтересованных...

...

Как мы это сделали

- Делаем анализаторы более 18 лет

Как мы это сделали

- Делаем анализаторы более 18 лет

- Go-анализатор из первых 40 диагностических правил

Как мы это сделали

- Делаем анализаторы более 18 лет
- Go-анализатор из первых 40 диагностических правил
- Тестер с размеченными срабатываниями из 33 проектов

Как мы это сделали

- Делаем анализаторы более 18 лет
- Go-анализатор из первых 40 диагностических правил
- Тестер с размеченными срабатываниями из 33 проектов
- Стресс тестер из ~1000 проектов для статистики

Но об этом не сегодня...
Смотрим ошибки!

Универсальные ошибки

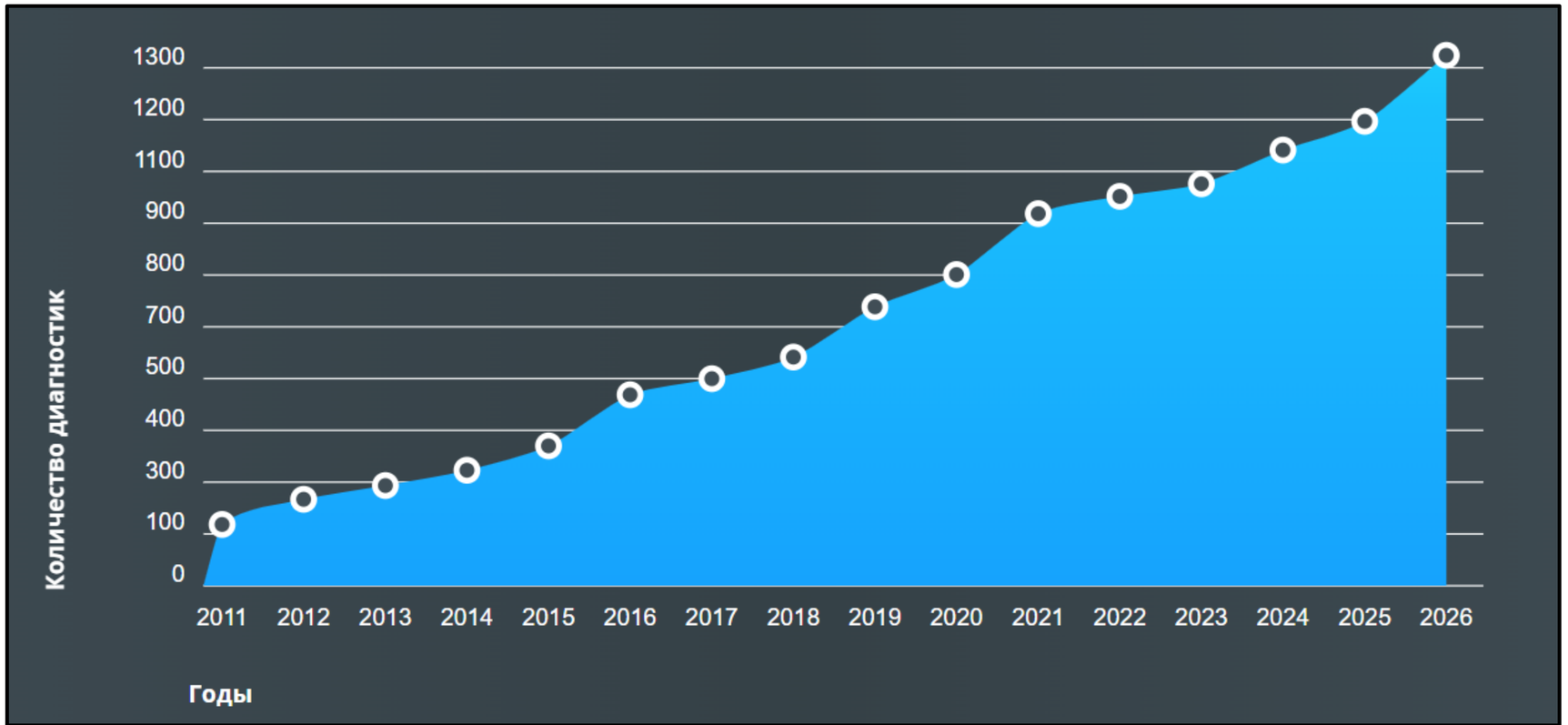
- Достаточно простые, но все такие же серьезные

Универсальные ошибки

- Достаточно простые, но все такие же серьезные
- Совершить может каждый и буквально везде

Универсальные ошибки

- Достаточно простые, но все такие же серьезные
- Совершить может каждый и буквально везде
- У нас в этом большой опыт!



Пример. Синтетический


```
if A == B {
   if A == B {
      ....
   }
}
```

Пример. Синтетический


```
if A == B {
    if A == B {
        ....
    }
}
```

Сообщение PVS-Studio:

V8020. Recurring check. This condition was already verified on a previous line.

- Ха-ха, ошибка слишком простая! Вот я точно так не ошибусь!

Пример. Синтетический


```
if A == B {
    if A == B {
        ....
    }
}
```

Сообщение PVS-Studio:

V8020. Recurring check. This condition was already verified on a previous line.

- Ха-ха, ошибка слишком простая! Вот я точно так не ошибусь!

...Ведь...так?

Пример. Из проекта Incus


```
err = p.Save(pidPath)
if err != nil {
    err2 := p.Stop()
    if err != nil {
        return fmt.Errorf("...: %s: %s", err, err2)
    }
    ...
}
```

Пример. Из проекта Incus


```
err = p.Save(pidPath)
if err != nil {
    err2 := p.Stop()
    if err != nil {      // <= похоже, должно быть err2
        return fmt.Errorf("...: %s: %s", err, err2)
    }
    ...
}
```

Сообщение PVS-Studio:

V8020 Recurring check. The 'err != nil' condition was already verified on line 407

Пример. Из проекта Incus


```
err = p.Save(pidPath)
if err != nil {
    err2 := p.Stop()
    if err != nil {       // <= похоже, должно быть err2
        return fmt.Errorf("...: %s: %s", err, err2)
    }
    ...
}
```

Пример. Синтетический


```
func Translate(value *T, numericValue int) {
    numericValue = reader.ReadInt32()
    ...
}
```

Пример. Синтетический


```
func Translate(value *T, numericValue int) {
    numericValue = reader.ReadInt32() // <= исходное
    ...                               значение
}                                     парпаметра
                                      нигде не
                                      используется
```

Сообщение PVS-Studio:

V8038. A parameter is always rewritten in the function body before being used.

Пример. Из проекта Kubernetes


```
func (c *FakeObjectConvertor)
   Convert(in, out, context interface{}) error {
      out = in
      return nil
}
```

Пример. Из проекта Kubernetes


```
func (c *FakeObjectConvertor)
   Convert(in, out, context interface{}) error {
      out = in
      return nil
}
```

Сообщение PVS-Studio:

V8038 The 'out' parameter is always rewritten in the function body before being used.

Пример. Синтетический


```
if l >= 0x06C0 && l <= 0x06CE {
  return true
}
if l == 0x06D5 {
  return true
}
if l >= 0x0905 && l <= 0x0939 {
  return true
}
if l == 0x06D5 {
  return true
}
if l >= 0x0985 && l <= 0x098C {
  return true
}
```

Пример. Синтетический


```
if l >= 0x06C0 && l <= 0x06CE {
  return true
}
if l == 0x06D5 { // <=
  return true
}
if l >= 0x0905 && l <= 0x0939 {
  return true
}
if l == 0x06D5 { // <=
  return true
}
if l >= 0x0985 && l <= 0x098C {
  return true
}
```

Пример. Из проекта vault


```
func (c *Core) PersistTOTPKey(...) error {
    val, err := jsonutil.EncodeJSON(ks)
    if err != nil {
        return err
    }
    if c.barrier.Put(ctx, &logical.StorageEntry{...}); err != nil {
        return err
    }
    return nil
}
```

Пример. Из проекта vault


```
func (c *Core) PersistTOTPKey(...) error {
    val, err := jsonutil.EncodeJSON(ks)
    if err != nil {
        return err
    }
    if c.barrier.Put(ctx, &logical.StorageEntry{...}); err != nil {
        return err                                      // ^= проверяют
    }                                                   старый err
    return nil
}
```

Сообщение PVS-Studio:

V8014 Two 'if' statements have identical conditions. The first 'if' statement contains function return, making the second 'if' redundant, or the code contains a logical error.

Пример. Из проекта vault


```
func (c *Core) PersistTOTPKey(...) error {
    val, err := jsonutil.EncodeJSON(ks)
    if err != nil {
        return err
    }
    if c.barrier.Put(ctx, &logical.StorageEntry{...}); err != nil {
        return err
    }
    return nil
}
```

```
type Storage interface {
  ....
  Put(context.Context, *StorageEntry) error
  ....
}
```

Trojan Source


```
isAuthorized := false
/* verify before transfer */ if isAuthorized {
    TransferFunds(account, amount)
/* end verify */ }
```

Trojan Source


```
isAuthorized := false
/* verify before transfer */ if isAuthorized {
    TransferFunds(account, amount)
/* end verify */ }


isAuthorized := false
if isAuthorized {
    TransferFunds(account, amount)
```

Trojan Source


```
isAuthorized := false
/* verify before transfer */ if isAuthorized {
    TransferFunds(account, amount)
/* end verify */ }


isAuthorized := false
TransferFunds(account, amount)
```

Trojan Source


```
isAuthorized := false
/* verify before transfer */ if isAuthorized {
    TransferFunds(account, amount)
/* end verify */ }
```

```
isAuthorized := false
/*[RLO] } [LRI] if isAuthorized[PDI] [LRI] verify
before transfer */
    TransferFunds(account, amount)
/* end verify [RLO]{ [LRI]*/
```

Trojan Source

- `[LRI] if isAuthorized[PDI]`
  фрагмент между `LRI` и `PDI` изолируется и отображается слева направо:
  `if isAuthorized`

- `[LRI] verify before transfer */`
  до конца строки, тоже изолированный блок:
  `` `verify before transfer */` ``

- `[RLO]`
  разворачивает всё справа налево

Trojan Source

- [LRI] if isAuthorized[PDI]
  фрагмент между LRI и PDI изолируется и отображается слева направо:
  if isAuthorized

- [LRI]
  ДО КОН
  `verify before transfer */`

```
'verify before transfer */', 'if isAuthorized',
{пробел}, '{', {пробел}
```

- [RLO]
  разворачивает всё справа налево

Trojan Source

- [LRI] if isAuthorized[PDI]
  фрагмент между LRI и PDI изолируется и отображается слева направо:
  if isAuthorized

- [LRI]
  ДО КОН
  `verify before transfer */`

```
'verify before transfer */', 'if isAuthorized',
{пробел}, '{', {пробел}
```

Сообщение PVS-Studio:

- **V5901. OWASP. Code contains invisible characters that may alter its logic. Consider enabling the display of invisible characters in the code editor.**

Так это и go vet поймает!

Хватит ли...?


```
func rgb1(r float32, g float32, b float32) {
  if r > 1 || g > 1 || r > 1 {
....
  }
}
```

Хватит ли...?


```
func rgb1(r float32, g float32, b float32) {
  if r > 1 || g > 1 || r > 1 {
....
  }
}
```

Конечно хватит!

go vet: `redundant or: r > 1 || r > 1`

Хватит ли...?


```
func rgb1(r float32, g float32, b float32) {
  if r > 1 || g > 1 || 1 < r {
....
  }
}
```

А если так...

Хватит ли...?


```
func rgb1(r float32, g float32, b float32) {
  if r > 1 || g > 1 || 1 < r {
....
  }
}
```

А если так...

Конечно хватит!

Хватит ли...?


```
func rgb1(r float32, g float32, b float32) {
  if r > 1 || g > 1 || 1 < r {
....
  }
}
```

А если так...

Конечно хватит!

Ведь...так?


go vet не поможет

Хватит ли...?


```
func rgb1(r float32, g float32, b float32) {
  if r > 1 || g > 1 || 1 < r {
....
  }
}
```

А если так...

Конечно **не** хватит!

go vet: All good! 👍

А если не синтетика? Nuclei


```
func NewEntityParser(dir string) (*EntityParser, error) {
  cfg := &packages.Config{
    Mode:
                  packages.NeedName   | packages.NeedFiles |
                  packages.NeedImports| packages.NeedTypes |
                  packages.NeedSyntax | packages.NeedTypes |
                  packages.NeedModule | packages.NeedTypesInfo,
    ....
  }
}
```

А если не синтетика? Nuclei


```
func NewEntityParser(dir string) (*EntityParser, error) {
  cfg := &packages.Config{
    Mode:
                  packages.NeedName   | packages.NeedFiles |
                  packages.NeedImports| packages.NeedTypes |
                  packages.NeedSyntax | packages.NeedTypes |
                  packages.NeedModule | packages.NeedTypesInfo,
    ....
  }
}
```

go vet: All good! 

Ну это наверное единичный случай...

Как перестараться и недоделать одновременно

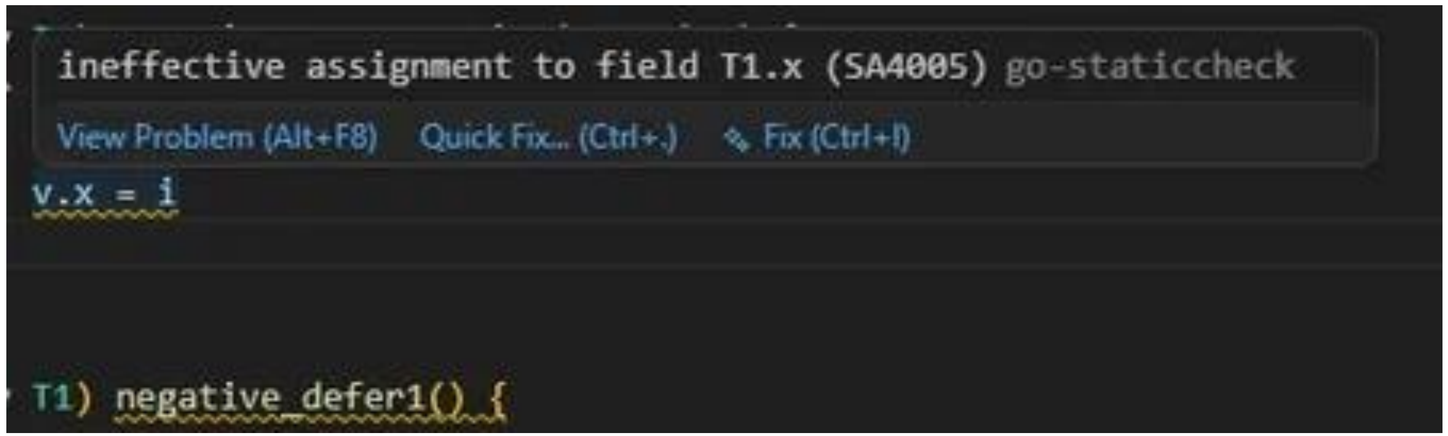
- Staticchek

```
func (v T1) negative_range_string(s string) {
    for i := range s {
        fmt.Println(v.x)
        v.x = i
    }
}
```

Как перестараться и недоделать одновременно

- Staticchek

```
func (v T1) negative_range_string(s string) {
    for i := range s {
        fmt.Println(v.x)
        v.x = i
    }
}
```



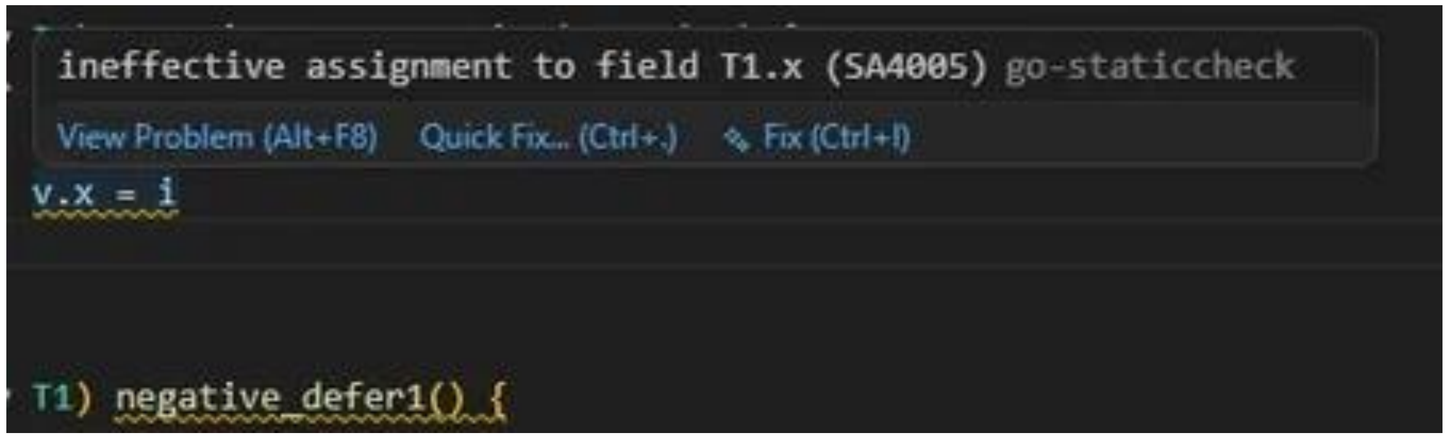
Как перестараться и недоделать одновременно

- Staticchek

```
func (v T1) negative_range_string(s string) {
    for i := range s {
        fmt.Println(v.x)
        v.x = i
    }
}
```

Но ошибка ли это?

Нет!



Как перестараться и недоделать одновременно

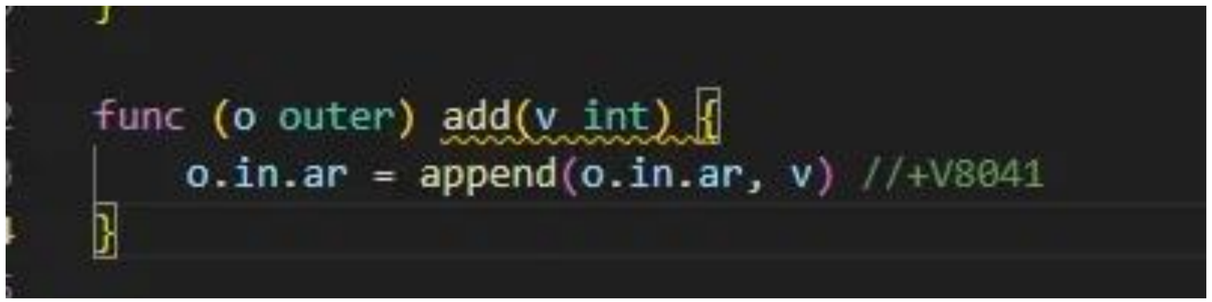
- Еще Staticchek

```
func (o outer) add(v int) {
    o.in.ar = append(o.in.ar, v)
}
```

Как перестараться и недоделать одновременно

- Еще Staticchek

```
func (o outer) add(v int) {
    o.in.ar = append(o.in.ar, v)
}
```



Но ошибка ли это?

Да!

Кто-то смог, а кто-то нет

- Еще пример

```
var v interface {
    Read()
}

switch v.(type) {
case io.ReadCloser: //+V8035
    fmt.Println(1)
default:
    fmt.Println(2)
}
```

Специфичные ошибки

XOR не такой каким кажется

- `^(XOR)` в Lua/VB.NET/R всё по классике и возводит в степень
- Но в Go `^(XOR)` не имеет отношения к возведению в степень

XOR не такой каким кажется

- `^(XOR)` в Lua/VB.NET/R всё по классике и возводит в степень
- Но в Go `^(XOR)` не имеет отношения к возведению в степень

На примере:

```
x := 2 ^ 16
```

Ожидание

```
x:= 65636 (2^16)
```

Реальность

```
x:=18
```

Пример. Из проекта Lancet


```
func (q *ArrayQueue[T]) Enqueue(item T) bool {
  if q.tail < q.capacity {
    q.data = append(q.data, item)
    // q.tail++
    q.data[q.tail] = item
  } else {
    //upgrade
    if q.head > 0 {
      ....
    } else {
      if q.capacity < 65536 {
        if q.capacity == 0 {
          q.capacity = 1
        }
        q.capacity *= 2
      } else {
        q.capacity += 2 ^ 16 // <=
      }
      tmp := make([]T, q.capacity, q.capacity)
      copy(tmp, q.data)
      q.data = tmp
    }
    q.data[q.tail] = item
  }
  q.tail++
  q.size++
  return true
}
```

Сообщение PVS-Studio:

V8015 Suspicious use of the bitwise XOR operator '^'. The exponentiation operation may have been intended here

Пример. Из проекта Lancet


```
if q.capacity < 65536 {
        if q.capacity == 0 {
          q.capacity = 1
        }
        q.capacity *= 2
      } else {
        q.capacity += 2 ^ 16 // <=
      }
```

Пример. Из проекта Lancet


```
func (q *ArrayQueue[T]) Enqueue(item T) bool {
  if q.tail < q.capacity {
    q.data = append(q.data, item)
    // q.tail++
    q.data[q.tail] = item
  } else {
    //upgrade
    if q.head > 0 {
      ....
    } else {
      if q.capacity < 65536 {
        if q.capacity == 0 {
          q.capacity = 1
        }
        q.capacity *= 2
      } else {
        q.capacity += 2 ^ 16 // <=
      }
      tmp := make([]T, q.capacity, q.capacity)
      copy(tmp, q.data)
      q.data = tmp
    }
    q.data[q.tail] = item
  }
  q.tail++
  q.size++
  return true
}
```

Сообщение PVS-Studio:

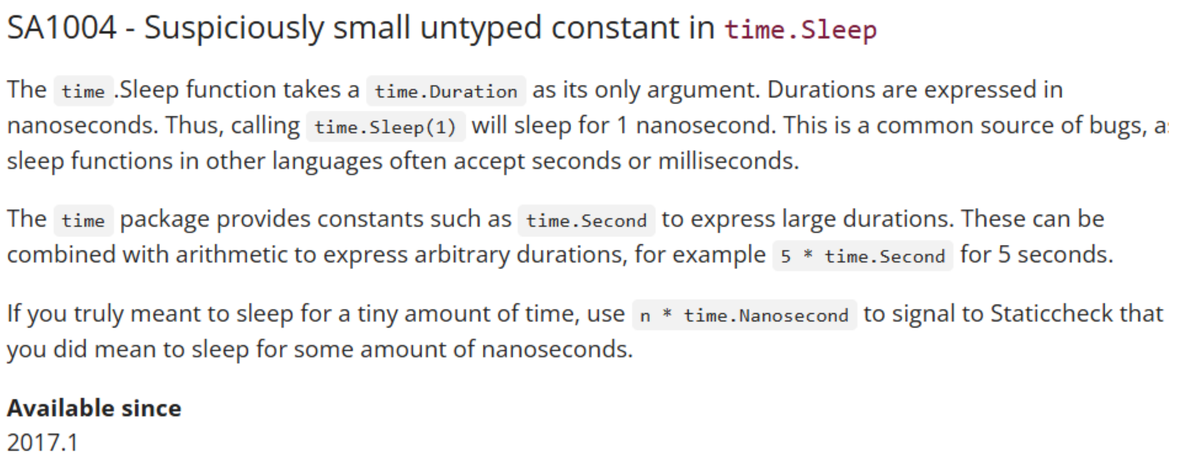
V8015 Suspicious use of the bitwise XOR operator '^'. The exponentiation operation may have been intended here

Документация

- Описание (из-за чего, откуда, оф. источники и доки)
- Пример ошибки
- Пример исправления (с объяснением как)
- Маппинг по стандартам
- Примеры из реальных проектов

Документация

- Описание (из-за чего, откуда, оф. источники и доки)
- Пример ошибки
- Пример исправления (с объяснением как)
- Маппинг по стандартам
- Примеры из реальных проектов



SA1004 - Suspiciously small untyped constant in `time.Sleep`

The `time`.Sleep function takes a `time.Duration` as its only argument. Durations are expressed in nanoseconds. Thus, calling `time.Sleep(1)` will sleep for 1 nanosecond. This is a common source of bugs, a sleep functions in other languages often accept seconds or milliseconds.

The `time` package provides constants such as `time.Second` to express large durations. These can be combined with arithmetic to express arbitrary durations, for example `5 * time.Second` for 5 seconds.

If you truly meant to sleep for a tiny amount of time, use `n * time.Nanosecond` to signal to Staticcheck that you did mean to sleep for some amount of nanoseconds.

Available since
2017.1

Надеюсь это всё... (нет)

На этом статический анализ не заканчивается

ГОСТ

- ГОСТ 71207 и 56939
- Для комплируемых языков – попадает под требования
- Список направлений на которые он распространяется
- Go vet из коробки не хватит

ИИ

- Объемы кода, отличающегося от человеческого (ошибки)
- Размеры инструментов и удобство за пределами IDE
- Агенты

А что делать?

PVS-Studio 8.0

- Поддержка анализа Go, Js, Ts
  (помимо анализаторов C\C++\C#\Java)
- По 40 новых диагностик для каждого языка
  (помимо более 1000 существующих)
- Новые IDE и обновление классических
  (масштабная переработка VS Code)

Q/A