Уязвимость регулярных выражений

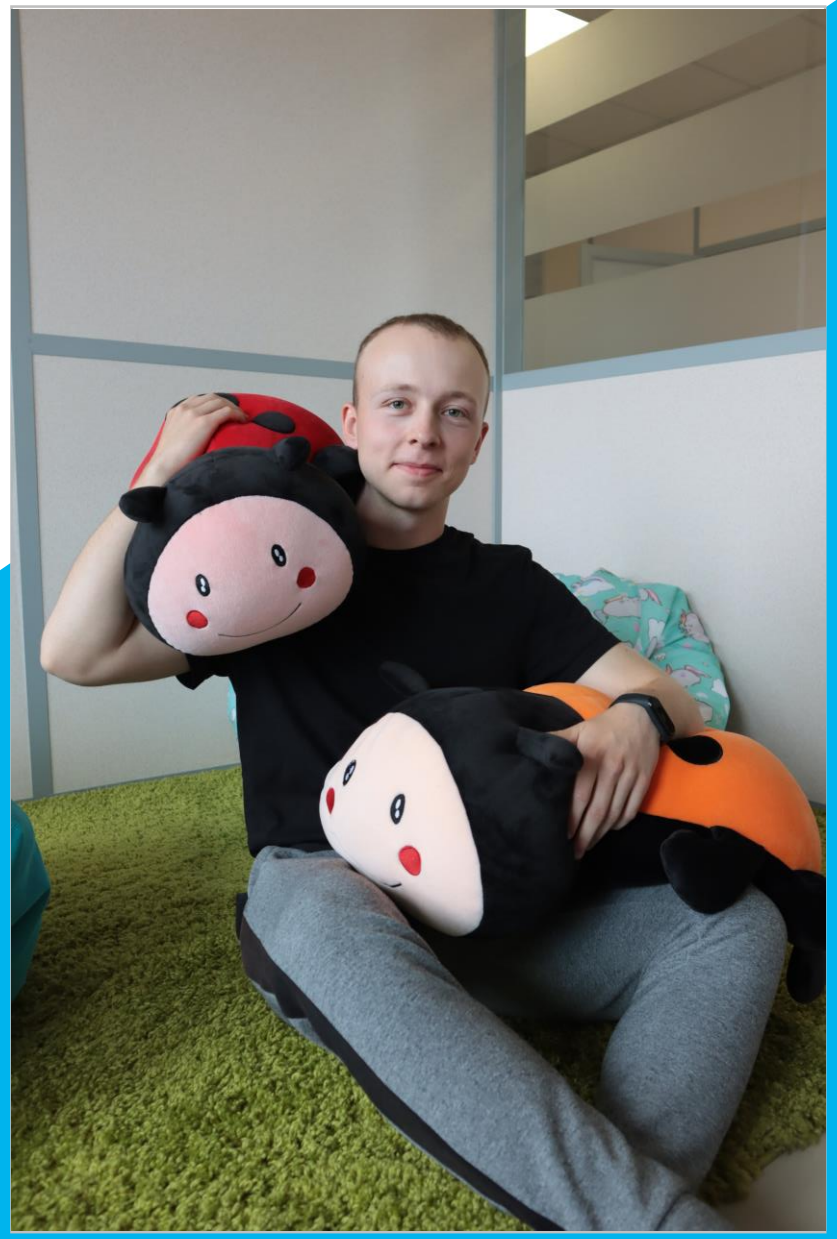
Теория и практика ReDoS-атак

Алексей Авдеев
avdeev@viva64.com

Speaker

Алексей Авдеев

- Разработчик в C# команде PVS-Studio
- Изучаю информационную безопасность
- Окружен багами, но не сломлен



Введение

Что за зверь такой, ReDoS?

ReDoS - regular expression denial of service

Разновидность DOS атаки, инициируемой при помощи уязвимого регулярного выражения

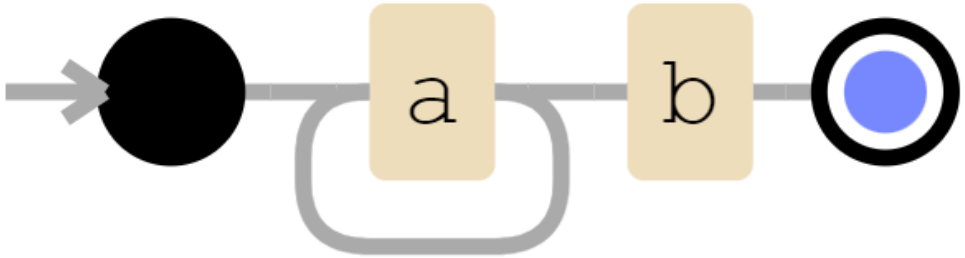
Чем опасен ReDoS?

- Замедление приложения
- Отказ в обслуживании

Рассмотрим пример

Регулярное выражение: a+b

Последовательность для разбора: aaa

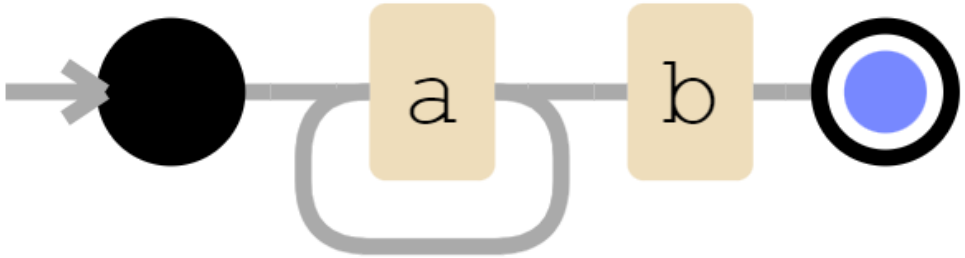


Номер шага	Последовательность для разбора	Сопоставление с паттерном a+	Сопоставление с паттерном b
1	aaa	aaa	Not Found

Рассмотрим пример

Регулярное выражение: a+b

Последовательность для разбора: aaa

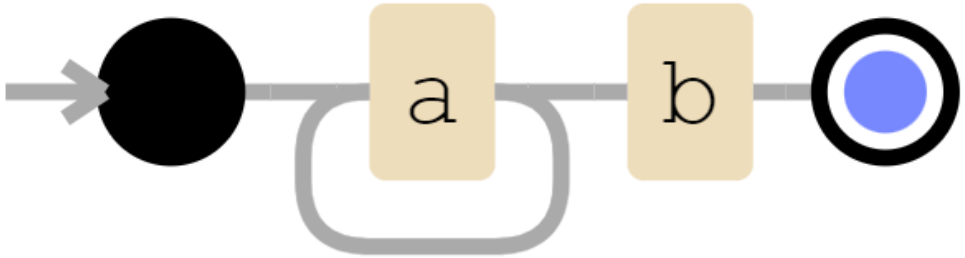


Номер шага	Последовательность для разбора	Сопоставление с паттерном a+	Сопоставление с паттерном b
1	aaa	aaa	Not Found
2	aaa	aa	Not Found

Рассмотрим пример

Регулярное выражение: a+b

Последовательность для разбора: aaa

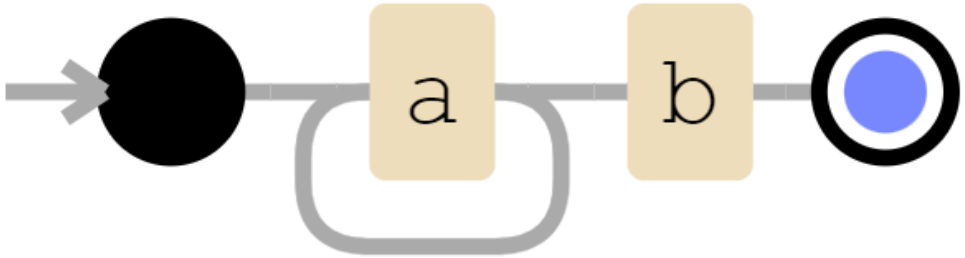


Номер шага	Последовательность для разбора	Сопоставление с паттерном a+	Сопоставление с паттерном b
1	aaa	aaa	Not Found
2	aaa	aa	Not Found
3	aaa	a	Not Found

Рассмотрим пример

Регулярное выражение: a+b

Последовательность для разбора: aaa

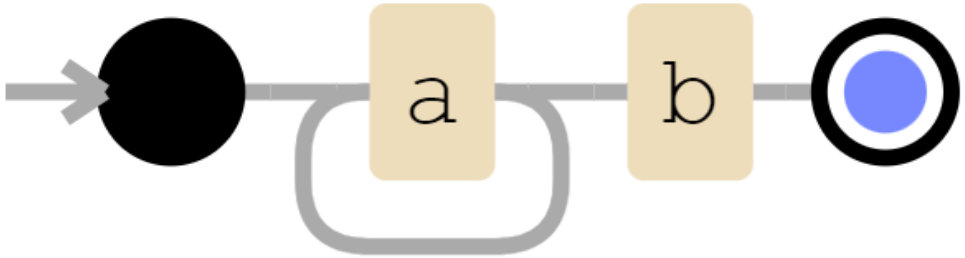


Номер шага	Последовательность для разбора	Сопоставление с паттерном a+	Сопоставление с паттерном b
1	aaa	aaa	Not Found
2	aaa	aa	Not Found
3	aaa	a	Not Found
4	aaa	aa	Not Found

Рассмотрим пример

Регулярное выражение: a+b

Последовательность для разбора: aaa

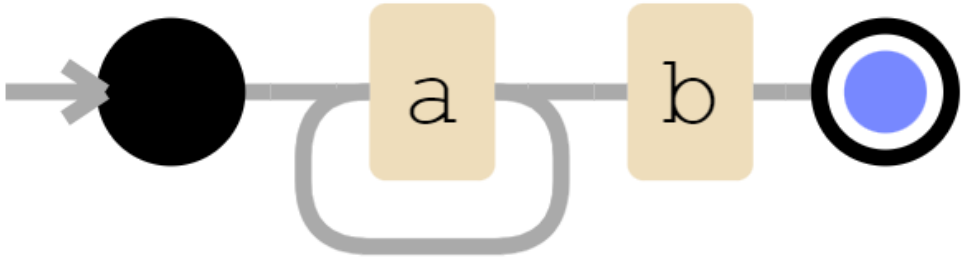


Номер шага	Последовательность для разбора	Сопоставление с паттерном a+	Сопоставление с паттерном b
1	aaa	aaa	Not Found
2	aaa	aa	Not Found
3	aaa	a	Not Found
4	aaa	aa	Not Found
5	aaa	a	Not Found

Рассмотрим пример

Регулярное выражение: a+b

Последовательность для разбора: aaa

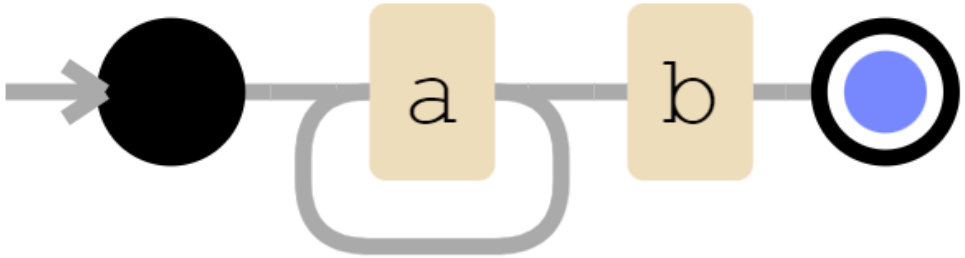


Номер шага	Последовательность для разбора	Сопоставление с паттерном a+	Сопоставление с паттерном b
1	aaa	aaa	Not Found
2	aaa	aa	Not Found
3	aaa	a	Not Found
4	aaa	aa	Not Found
5	aaa	a	Not Found
6	aaa	a	Not Found

Рассмотрим пример

Регулярное выражение: a+b

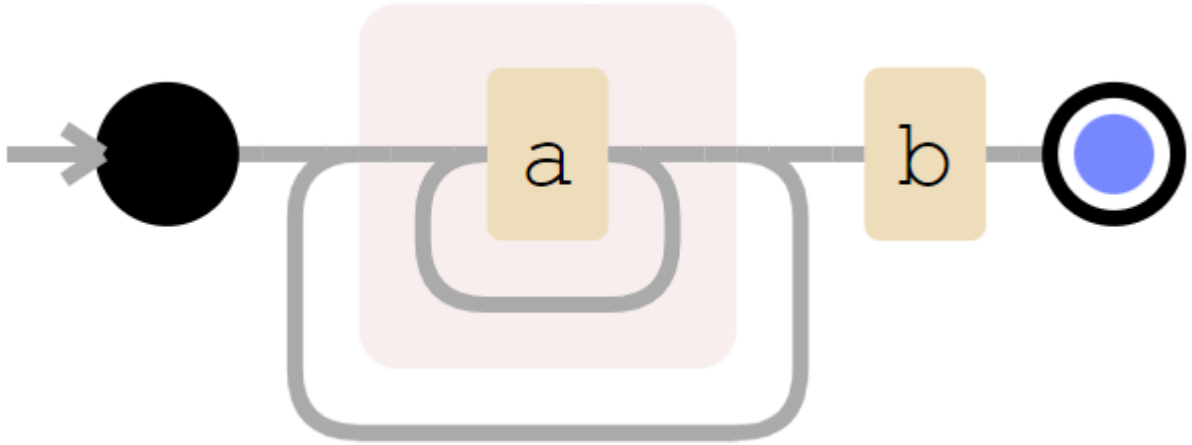
Последовательность для разбора: aaa



Номер шага	Последовательность для разбора	Сопоставление с паттерном a+	Сопоставление с паттерном b
1	aaa	aaa	Not Found
2	aaa	aa	Not Found
3	aaa	a	Not Found
4	aaa	aa	Not Found
5	aaa	a	Not Found
6	aaa	a	Not Found
7	aaa	Not Found	Not Found

Рассмотрим пример

Регулярное выражение: (a+)+b



Что изменилось?

- На графе появилась дополнительная петля
- Количество возможных вариантов сопоставления увеличилось
- Увеличилось время, требуемое для разбора последовательности

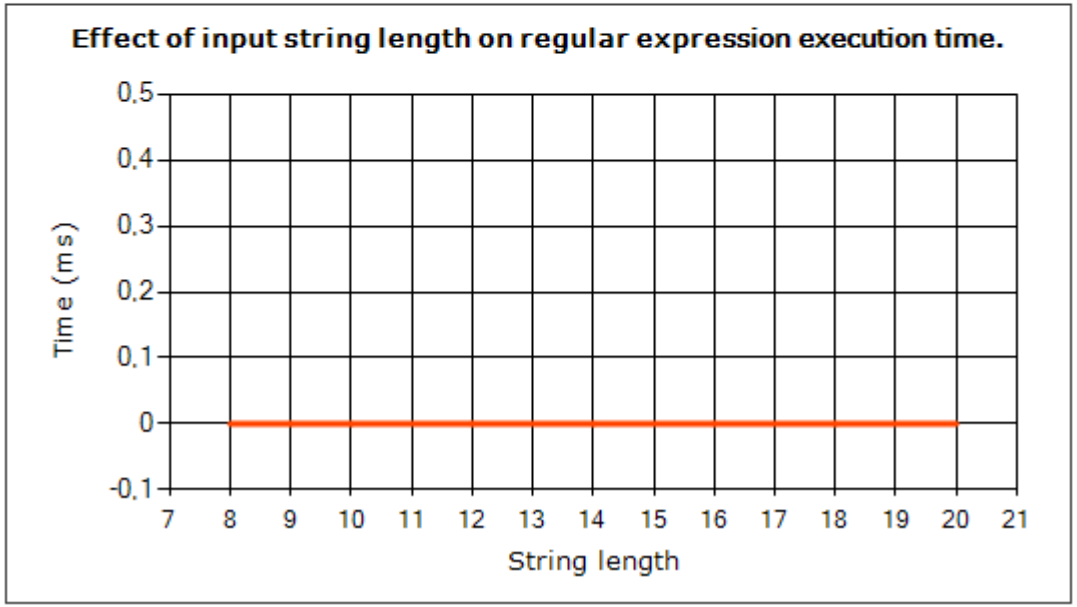
Или нет?

Сравним два регулярных выражения

a+b

Input strings:

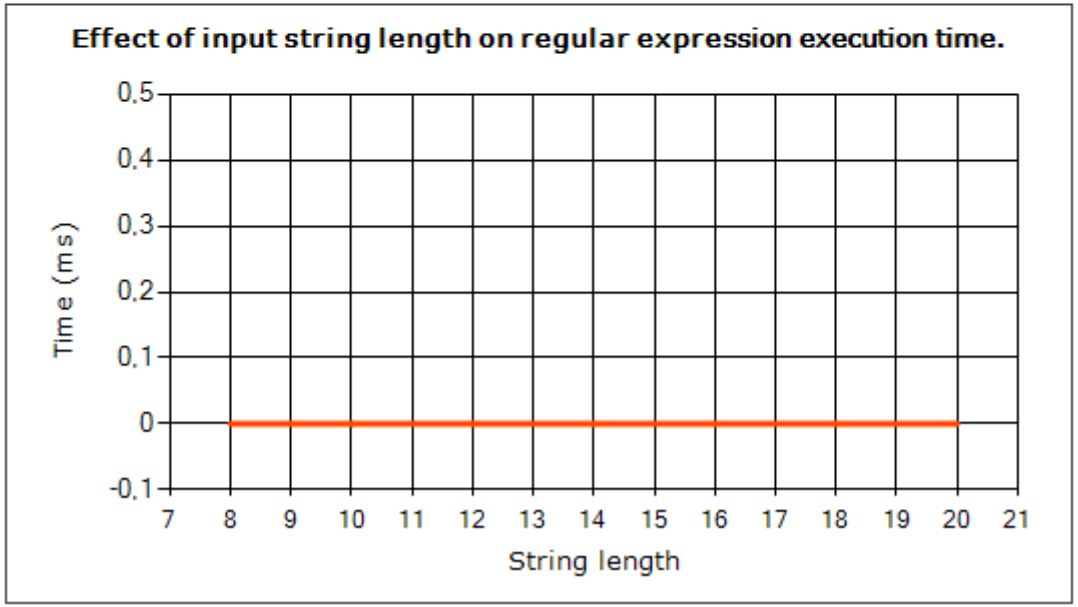
aaaaaaab
aaaaaaaab
aaaaaaaaab
aaaaaaaaaaab
aaaaaaaaaaaaab
aaaaaaaaaaaaaaab
aaaaaaaaaaaaaaaab
aaaaaaaaaaaaaaaaab



(a+)+b

Input strings:

aaaaaaab
aaaaaaaab
aaaaaaaaab
aaaaaaaaaaab
aaaaaaaaaaaaab
aaaaaaaaaaaaaaab
aaaaaaaaaaaaaaaab
aaaaaaaaaaaaaaaaab

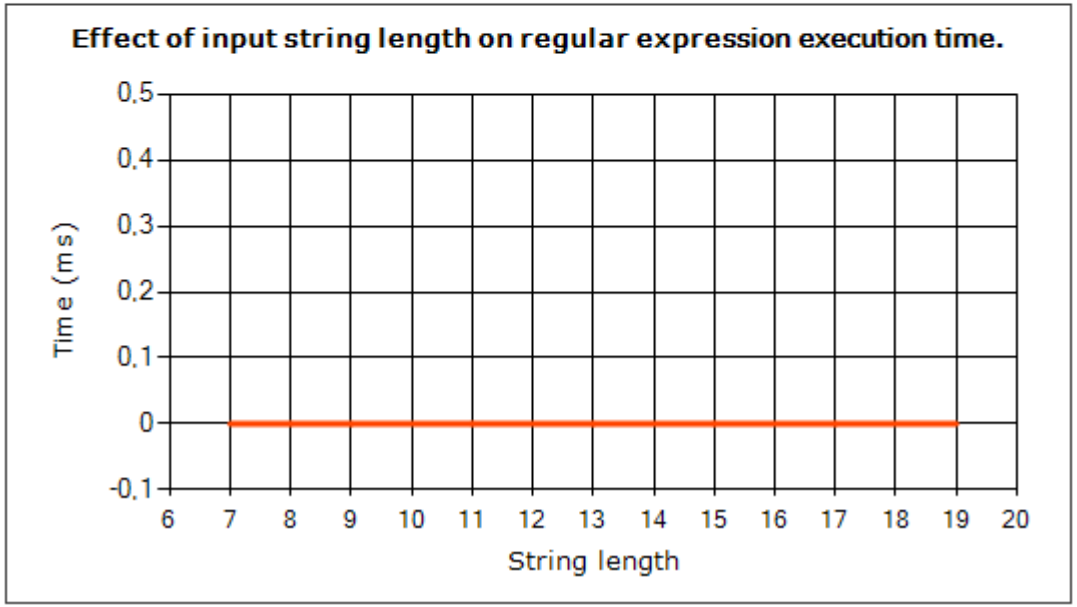


Еще один тест

a+b

Input strings:

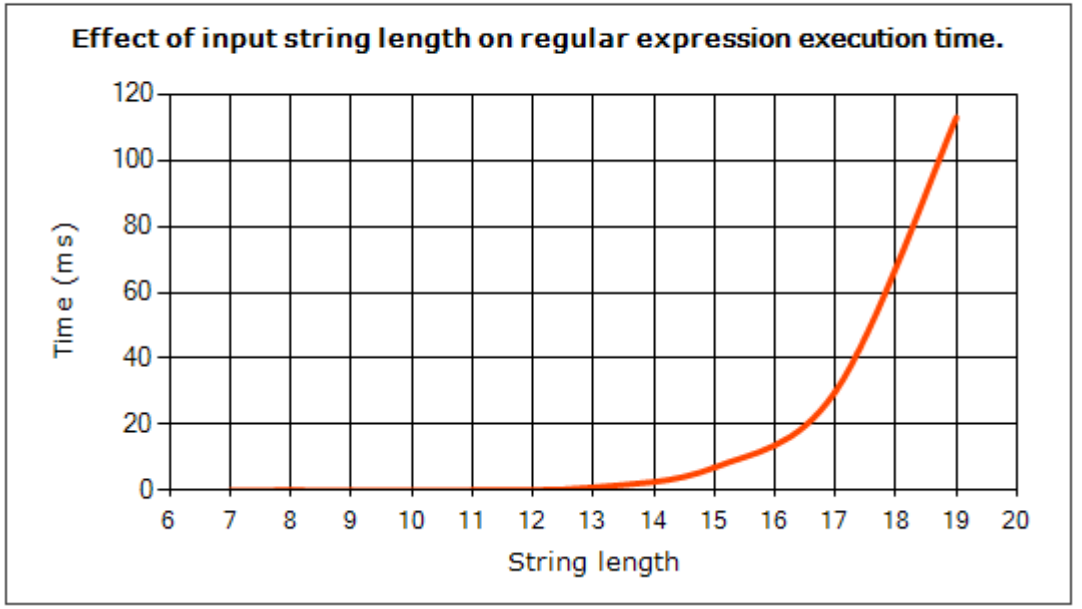
aaaaaaa
aaaaaaaa
aaaaaaaaa
aaaaaaaaaaa
aaaaaaaaaaaaa
aaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaa



(a+)+b

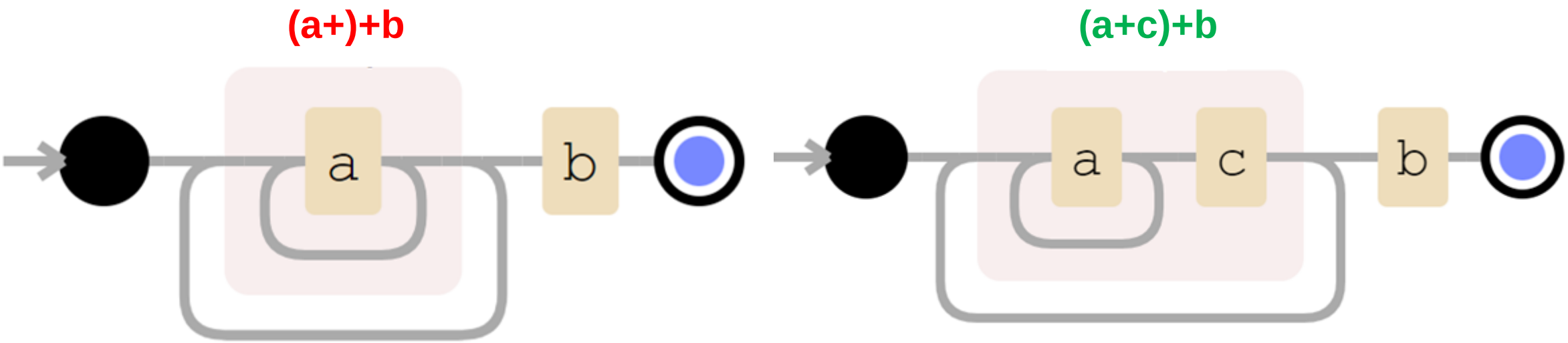
Input strings:

aaaaaaa
aaaaaaaa
aaaaaaaaa
aaaaaaaaaaa
aaaaaaaaaaaaa
aaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaa



Катастрофический возврат

- Существует два подвыражения, при этом одно из них включено в другое и к каждому из них применяется один из следующих кванторов: '*', '+', '*?', '+?', '{...}'
- Существует такая строка, которую можно было бы сопоставить с обоими этими подвыражениями



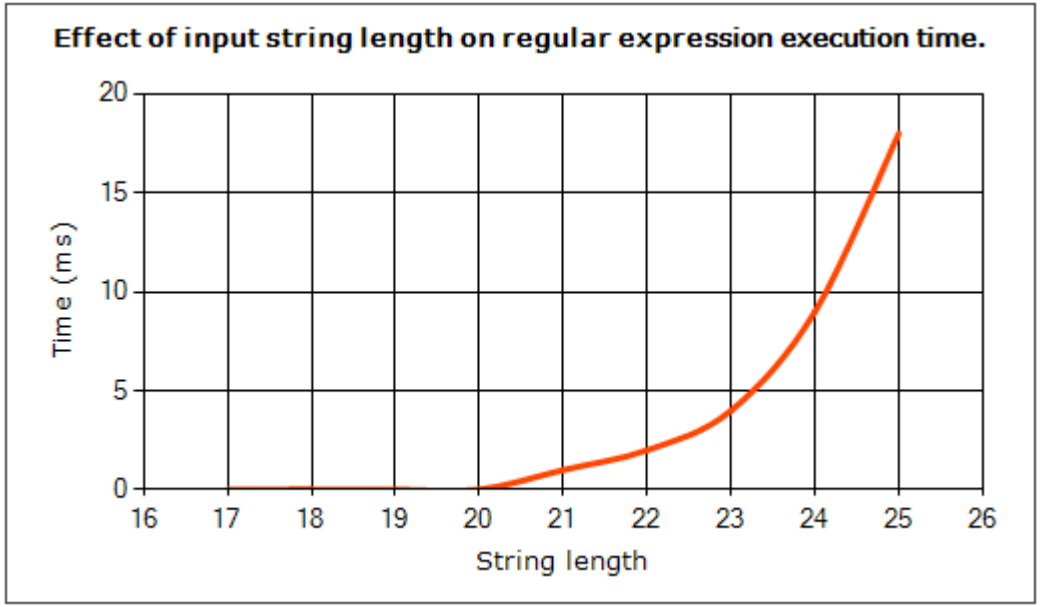
Уязвимости на реальных проектах

RestSharp

- CVE-2021-27293
- Regex: newDate\(**(-?\d+)***\)
- *Ищет строку формата: newDate(10-01-2007)*

Input strings:

```
newDate(111111111
newDate(1111111112
newDate(11111111133
newDate(111111111444
newDate(1111111115555
newDate(11111111166666
newDate(111111111777777
newDate(1111111118888888
newDate(11111111199999999
```



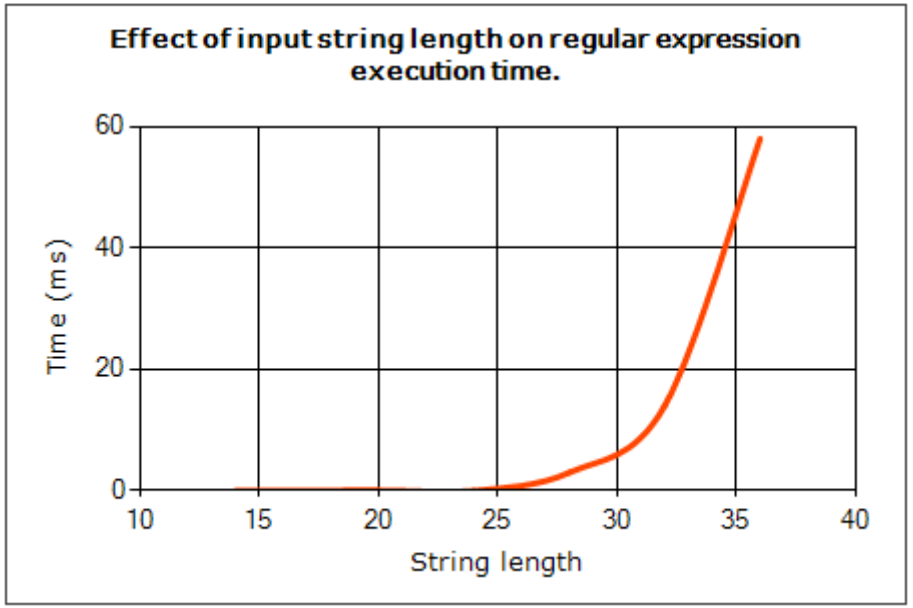
System.ComponentModel.DataAnnotations

- CVE-2015-2526
- Классы EmailAddressAttribute и PhoneAttribute
- Попытка валидации email и номера телефона

email validation

Input strings:

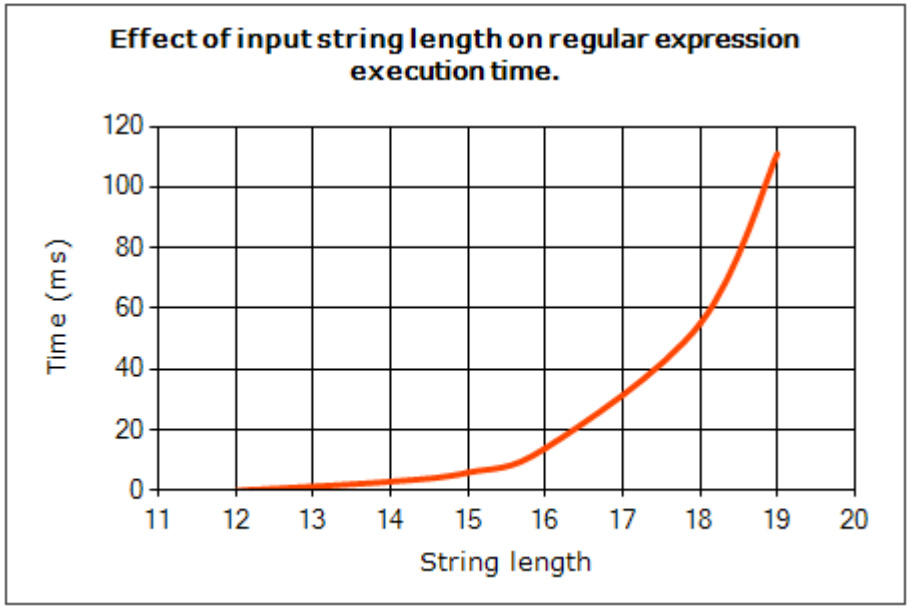
```
t@t.t.t.c%20
t@t.t.t.t.t.c%20
t@t.t.t.t.t.t.t.c%20
t@t.t.t.t.t.t.t.t.t.c%20
t@t.t.t.t.t.t.t.t.t.t.t.c%20
t@t.t.t.t.t.t.t.t.t.t.t.t.t.c%20
```



phone validation

Input strings:

```
66666666666d
6666666666666d
66666666666666d
666666666666666d
6666666666666666d
66666666666666666d
```



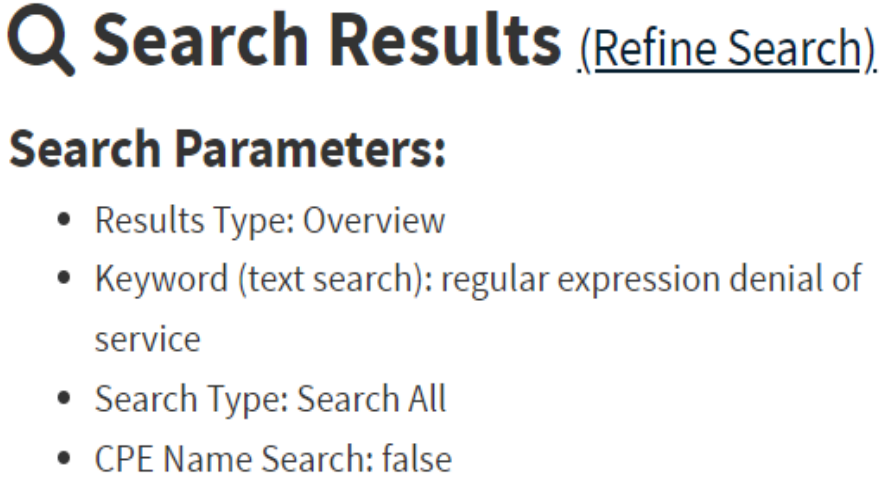
ReDoS вне C# кода

- **Cloudflare (2019)**

WAF правило содержало опасное регулярное выражение

- **Stack Overflow (2016)**

DOS в результате обработки опасным регулярным выражением вопроса, содержащего 20 000 символов пробела



Из них за:

- 2021: 66 уязвимостей
- 2022: 57 уязвимостей

Что делать, если у нас ReDoS?

Что делать, если у нас ReDoS?

- Используйте другое регулярное выражение
- Замените регулярное выражение рукописным обработчиком
- Разбивайте регулярные выражения
- Проверяйте источник данных. Не всем данным стоит доверять
- Будьте внимательнее к сгенерированным в runtime и основанным на пользовательском вводе регулярным выражениям

Пример ReDoS via injection

```
String login = loginTB.Text;
String password = passwordTB.Text;
Regex testPassword = new Regex(login);
Match match = testPassword.Match(password);
if (match.Success)
{
    MessageBox.Show("Don't include login in password.");
}
else
{
    MessageBox.Show("Good password.");
}
```

Пользовательский ввод:

login: (a*)*b

password: aaaaaaaaaaaaaaa!

Результат – ReDoS!

Что делать, если у нас ReDoS?

- Используйте другое регулярное выражение
- Замените регулярное выражение рукописным обработчиком
- Разбивайте регулярные выражения
- Проверяйте источник данных. Не всем данным стоит доверять
- Будьте внимательнее к сгенерированным в runtime и основанным на пользовательском вводе регулярным выражениям
- Используйте timeout

Используйте timeout

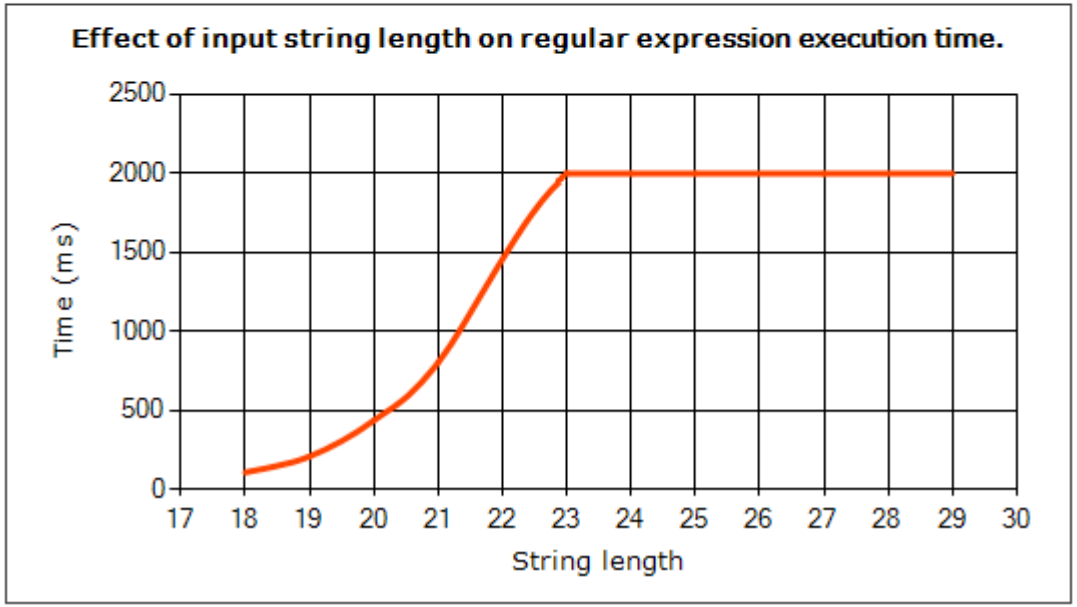
Третий аргумент конструктора регулярных выражений – matchTimeout

В случае превышения времени разбора выбросит RegexMatchTimeoutException

```
Regex regex = new Regex("(a*)*b",
                        RegexOptions.None,
                        TimeSpan.FromSeconds(2));
```

Input strings:

aaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaa



Что делать, если у нас ReDoS?

- Используйте другое регулярное выражение
- Замените регулярное выражение рукописным обработчиком
- Разбивайте регулярные выражения
- Проверяйте источник данных. Не всем данным стоит доверять
- Обратите внимание на сгенерированные в runtime и основанные на пользовательском вводе регулярным выражениям
- Используйте timeout
- Используйте атомарные группы

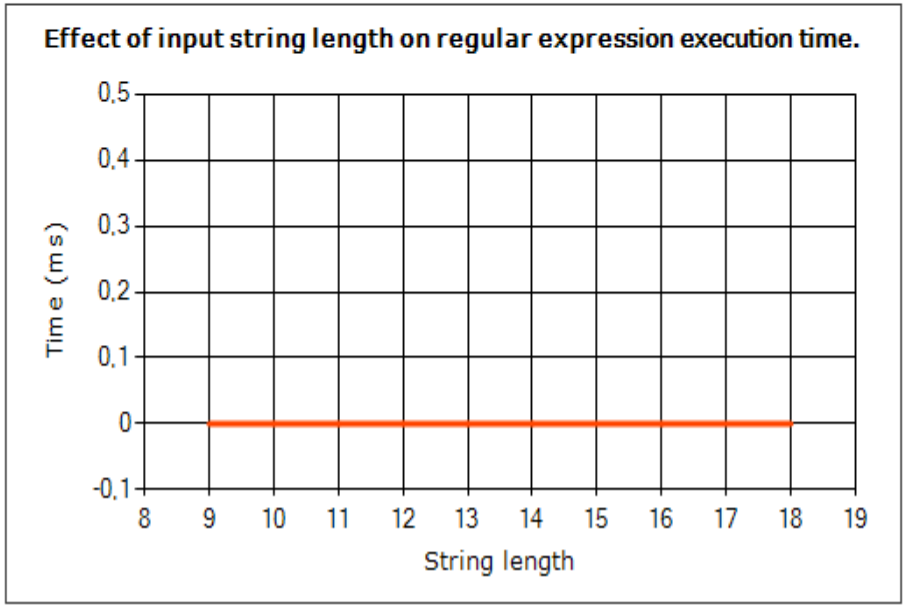
Используйте атомарные группы

- Синтаксис определения атомарных групп: (?>...)
- Для выражения внутри отключается функция обратного отслеживания
- Внимание! Использование атомарных групп меняет логику обработки соответствий

(?>a*)*b

Input strings:

```
aaaaaaaaa
aaaaaaaaaa
aaaaaaaaaaa
aaaaaaaaaaaa
aaaaaaaaaaaaa
aaaaaaaaaaaaaa
aaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa
```



В чем разница?

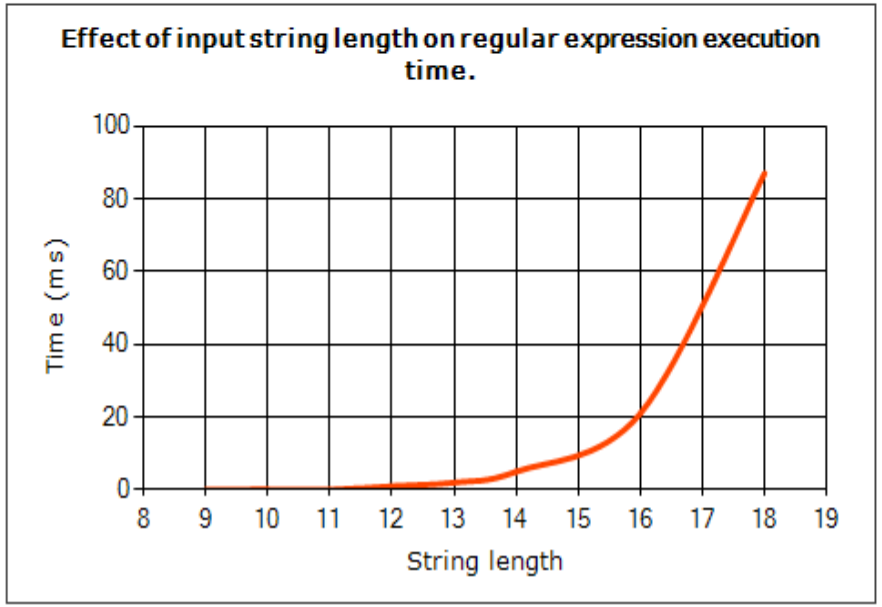
- Два одинаковых регулярных выражения
- Одинаковые входные данные
- Абсолютно одинаковый код
- Не фотошоп!

(a*)*b

Input strings:

aaaaaaaaa
aaaaaaaaaa
aaaaaaaaaaa
aaaaaaaaaaaa
aaaaaaaaaaaaa
aaaaaaaaaaaaaa
aaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa

.NET Framework 4.7.2

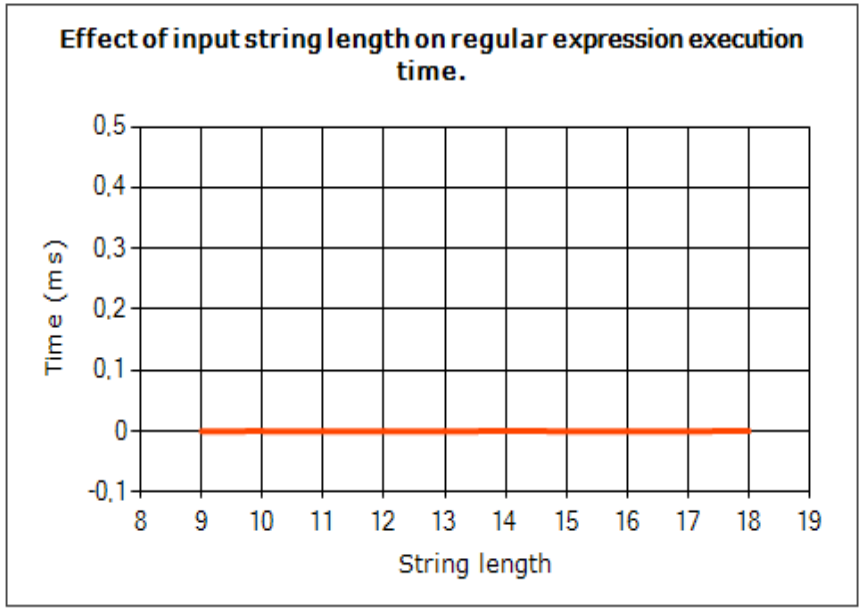


(a*)*b

Input strings:

aaaaaaaaa
aaaaaaaaaa
aaaaaaaaaaa
aaaaaaaaaaaa
aaaaaaaaaaaaa
aaaaaaaaaaaaaa
aaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa

.NET 5



Что делать, если у нас ReDoS?

- Используйте другое регулярное выражение
- Замените регулярное выражение рукописным обработчиком
- Разбивайте регулярные выражения
- Проверяйте источник данных. Не всем данным стоит доверять
- Обратите внимание на сгенерированные в runtime и основанные на пользовательском вводе регулярным выражениям
- Используйте timeout
- Используйте атомарные группы
- Используйте новые версии .NET
- Используйте движок Regex на основе DFA

DFA Regex engine

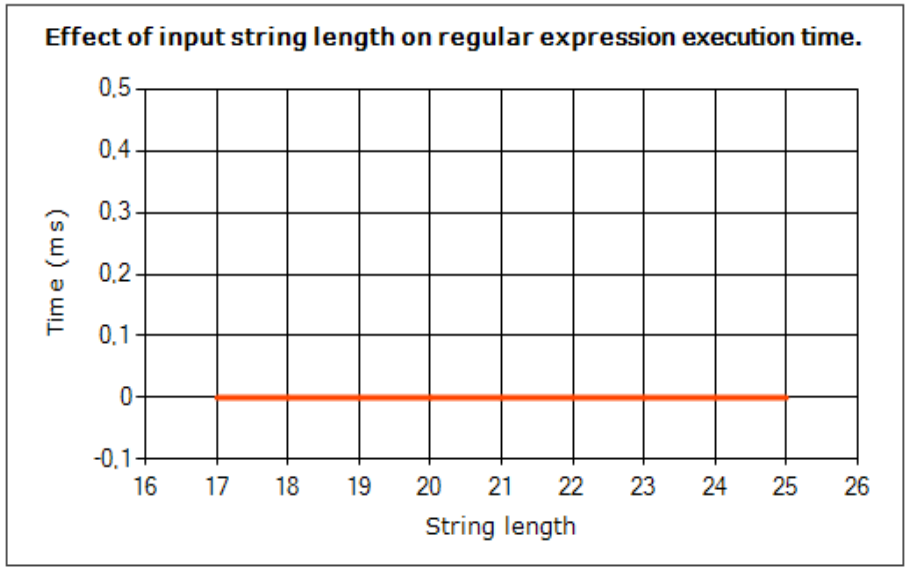
В .NET 7 вы можете указать флаг RegexOptions.NonBacktracking

```
Regex regex = new Regex(@"newDate\((-?\d+)*\)",
                        RegexOptions.NonBacktracking);
```

В таком случае для разбора будет использован основанный на DFA движок регулярных выражений

Input strings:

```
newDate(111111111
newDate(1111111112
newDate(11111111133
newDate(111111111444
newDate(1111111115555
newDate(11111111166666
newDate(111111111777777
newDate(1111111118888888
newDate(11111111199999999
```



Резюмируем

- Используйте другое регулярное выражение
- Замените регулярное выражение рукописным обработчиком
- Разбивайте регулярные выражения
- Проверяйте источник данных. Не всем данным стоит доверять
- Обратите внимание на сгенерированные в runtime и основанные на пользовательском вводе регулярным выражениям
- Используйте timeout
- Используйте атомарные группы
- Используйте новые версии .NET
- Используйте движок Regex на основе DFA

Профилактика ReDoS уязвимости

Профилактика ReDoS уязвимости

- Уделяйте время проверке чужих регулярных выражений

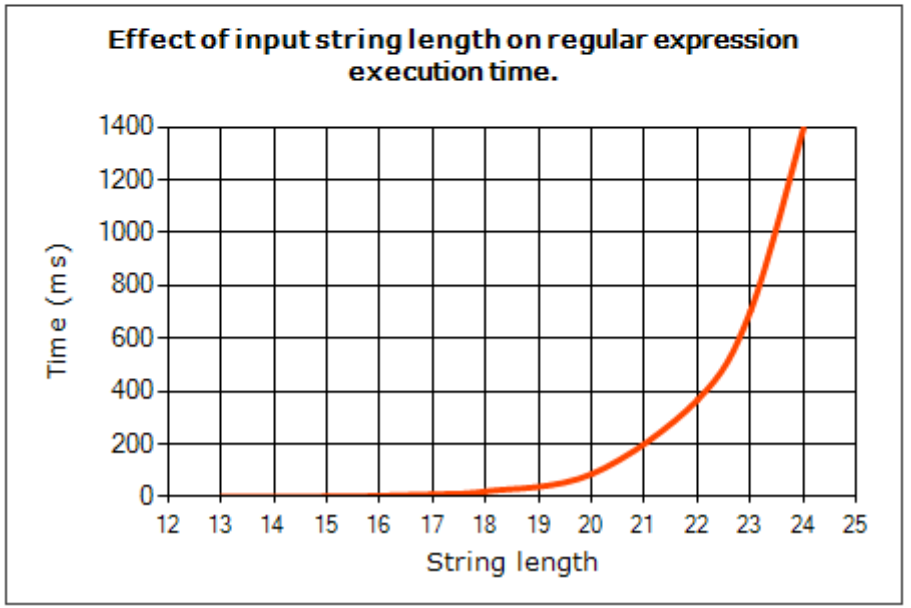
Доверяй, но проверяй

Regular Expression Details

Title	**RegEx for email validation** [Find] [Test]
Expression	/^([a-zA-Z0-9])(([\-.]|[_]+)?([a-zA-Z0-9]+))*(@){1}[a-z0-9]+[.]{1}(([a-z]{2,3})|([a-z]{2,3}[.]{1}[a-z]{2,3}))$/
Description	This expresssion will validate all possible formats except if web site URL contains hyphen characters like aa@a-b-c.com. I will include this feature also in next version.
Matches	a@abc.com,a@abc.co.in,aa.bb@abc.com.sg,aa_bb@abc.biz,aa.C.bb@abc.com,aa_1900@abc.co.in
Non-Matches	@abc.com,a@abc.co.in.in,a@abc.com.in.in,a@a-b-c.com
Author	Rafiq **Rating:**
Source	Email
Your Rating	**Bad** ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 **Good** [Submit Rating]

Input strings:

```
aaaaaaaaaaaa!
aaaaaaaaaaaaaa!
aaaaaaaaaaaaaaa!
aaaaaaaaaaaaaaaa!
aaaaaaaaaaaaaaaaa!
aaaaaaaaaaaaaaaaaa!
aaaaaaaaaaaaaaaaaaa!
aaaaaaaaaaaaaaaaaaaa!
aaaaaaaaaaaaaaaaaaaaa!
```



Профилактика ReDoS уязвимости

- Уделяйте время проверке чужих регулярных выражений
- Используйте утилиты для проверки безопасности регулярных выражений
- Обновляйте уязвимые зависимости
- Используйте SAST решения для анализа кода

Q&A

pvs-studio.com
avdeev@viva64.com