

# CookBook по регулярным выражениям (REGEX)

Проверка работы регулярок (выставить флаги gm):

- <https://regexr.com/>
- <https://regex101.com/>

Доп чтение:

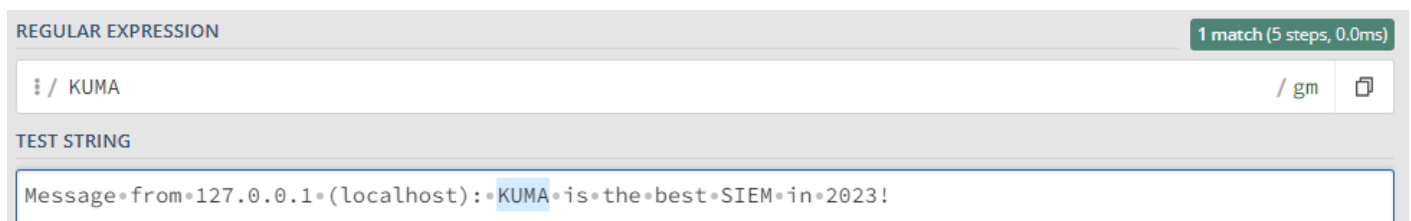
- <https://habr.com/ru/articles/545150/>
- [https://regex.sorokin.engineer/ru/latest/regular\\_expressions.html](https://regex.sorokin.engineer/ru/latest/regular_expressions.html)

Простейшие приемы, практику отработаем на тестовом сообщении:

Message from 127.0.0.1 (localhost): KUMA is the best SIEM in 2023!

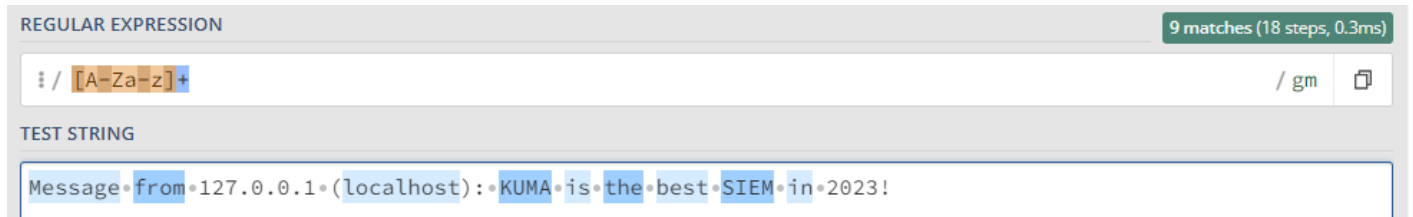
## Захватить строку KUMA

**KUMA** Ищется полное соответствие строке KUMA.



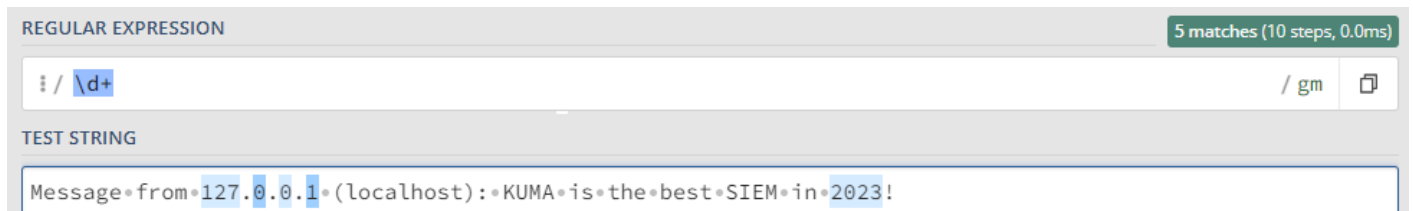
## Захватить строку содержащую только буквы

**[A-Za-z]+** Ищем группу (**[ ]**) символов с большими (**A-Z**) и маленькими (**a-z**) буквами от одной и более (+).



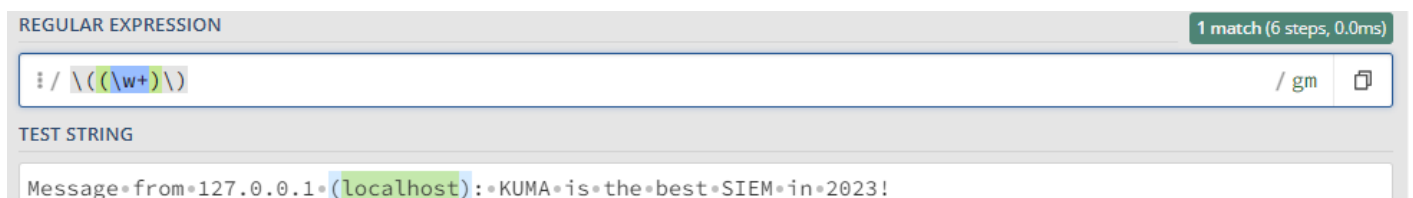
## Захватить строку содержащую только числа

**\d+** Ищем по токenu **\d**, что является эквивалентом **[0-9]** от одного и более вхождений (+).



## Захватить строку внутри круглых скобок

**\((\w+)\)** Ищем по токenu **\w**, что является эквивалентом **[a-zA-Z0-9\_]** от одного и более вхождений (+), при этом экранируем круглые скобки с помощью обратного следа **\** и строку нашу определяем в группу круглыми скобками (**()**)



## Захватить строку до двоеточия

`^[^\:]+` Ищем с начала строки `^`, далее захватываем в группе все кроме двоеточия (символ двоеточия экранирован) `[\^\:]` от одного и более вхождений (`+`)

REGULAR EXPRESSION 1 match (4 steps, 0.1ms)

`/^[^\:]+` / gm

TEST STRING

Message•from•127.0.0.1•(localhost):•KUMA•is•the•best•SIEM•in•2023!

## Захватить строку после двоеточия

`^[^\:]+$` Такая, подобная представленной выше, конструкция не подойдет, т.к. она будет очень емокой (633 шага). Ищем все кроме двоеточия (символ двоеточия экранирован) `[\^\:]` от одного и более вхождений (`+`), но до конца строки `$`

REGULAR EXPRESSION 1 match (633 steps, 0.1ms)

`/[\^\:]+$` / gm

TEST STRING

Message•from•127.0.0.1•(localhost):•KUMA•is•the•best•SIEM•in•2023!

В нашем случае лучше использовать следующее

`\:(.*)$` Ищем в строке двоеточие `\:`, далее захватываем все символы от нуля и более вхождений (`*`), и берем все что нам нужно в группу (`()`)

REGULAR EXPRESSION 1 match (6 steps, 0.0ms)

`/\:(.*)$` / gm

TEST STRING

Message•from•127.0.0.1•(localhost):•KUMA•is•the•best•SIEM•in•2023!

## Захватить IP-адрес

`\d+\.\d+\.\d+\.\d+` Ищем числа от одного и более `\d+`, с точкой и так 4 раза

REGULAR EXPRESSION 1 match (8 steps, 0.0ms)

`/\d+\.\d+\.\d+\.\d+/` / gm

TEST STRING

Message from 127.0.0.1 (localhost): KUMA is the best SIEM in 2023!

## Захватить слова состоящие из 4 букв

`\b[a-zA-Z]{4}\b` Ищем группу 4 символов из букв и разграничиваем их (boundary) `\b`

REGULAR EXPRESSION 4 matches (46 steps, 0.1ms)

`/\b[a-zA-Z]{4}\b/` / gm

TEST STRING

Message from 127.0.0.1 (localhost): KUMA is the best SIEM in 2023!

## Захватить слова состоящие от 3 до 4 букв

`\b[a-zA-Z]{3,4}\b` Ищем группу 4 символов из букв и разграничиваем их (boundary) `\b`

REGULAR EXPRESSION 5 matches (48 steps, 0.1ms)

`/\b[a-zA-Z]{3,4}\b/` / gm

TEST STRING

Message from 127.0.0.1 (localhost): KUMA is the best SIEM in 2023!

## Захватить IPv4 адрес

`\b((2([0-4][0-9]|5[0-5])|([0-1]?[0-9]?[0-9]))\.){3}((2([0-4][0-9]|5[0-5])|([0-1]?[0-9]?[0-9])))\b`

REGULAR EXPRESSION 3 matches (198 steps, 0.1ms)

```
:/\b((2([0-4][0-9]|5[0-5])|([0-1]?[0-9]?[0-9]))\.){3}((2([0-4][0-9]|5[0-5])|([0-1]?[0-9]?[0-9])))\b/ gm
```

TEST STRING

```
192.168.0.1
0.0.0.0
273.0.454.3
1.23.45.6
```

Более ленивый вариант, но быстрый, без валидации:

```
(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})
```

REGULAR EXPRESSION 4 matches (40 steps, 0.0ms)

```
:/(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})/ gm
```

TEST STRING

```
192.168.0.1
0.0.0.0
273.0.454.3
1.23.45.6
```

## Захватить IPv6 адрес

```
(([a-fA-F0-9]{1,4}|:){1,7}([a-fA-F0-9]{1,4}|:))
```

REGULAR EXPRESSION 5 matches (394 steps, 0.1ms)

```
:/((([a-fA-F0-9]{1,4}|:){1,7}([a-fA-F0-9]{1,4}|:)))/ gm
```

TEST STRING

```
The following list shows examples of valid IPv6 (Normal) addresses:
2001:db8:3333:4444:5555:6666:7777:8888.
2001:db8:3333:4444:CCCC:DDDD:EEEE:FFFF.
:: (implies all 8 segments are zero)
2001:db8:: (implies that the last six segments are zero)
::1234:5678 (implies that the first six segments are zero)
```

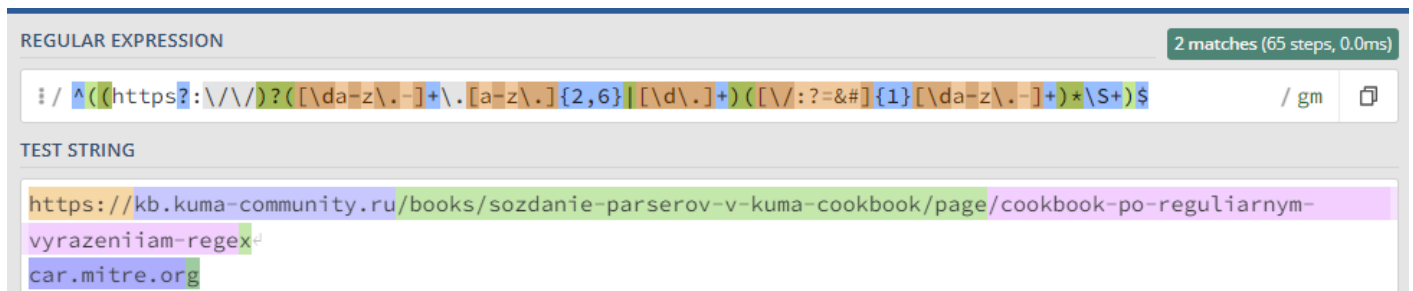
## Захватить HASH сумму

- MD `^[a-fA-F0-9]{32}$`
- SHA1 `^[a-fA-F0-9]{40}$`

- SHA256 `^[a-fA-F0-9]{64}$`
- SHA512 `^[a-fA-F0-9]{128}$`

## Захватить URL адрес

```
^((https?:\/\/)?([\da-z\.-]+\.[a-z\.-]{2,6}|[\d\.-]+)([\v:!?=&#]{1}[\da-z\.-]+)*\S+)$
```



## Захватить EMAIL адрес

Определяет почти все типы валидных адресов

```
\b[\w\.-\+!\/\\"%]+@[ \w\.-]+\([\w]{2,4}\)?\b
```

REGULAR EXPRESSION 16 matches (1 159 steps, 0.1ms)

:/ \b[\w\.-\+!\/\"%]+@[\w\.-]+\(\.\w{2,4}\)?\b / gm

TEST STRING

marketing@kuma-community.com  
 jessica-hr@kuma-community.com  
 very.common@example.com  
 x@example.com (one-letter local-part)  
 long.email-address-with-hyphens@and.subdomains.example.com  
 user.name+tag+sorting@example.com (may be routed to user.name@example.com inbox depending on mail server)  
 name/surname@example.com (slashes are a printable character, and allowed)  
 admin@example (local domain name with no TLD, although ICANN highly discourages dotless email addresses[29])  
 example@s.example (see the List of Internet top-level domains)  
 ""@example.org (space between the quotes)  
 "john..doe"@example.org (quoted double dot)  
 mailhost!username@example.org (bangified host route used for uucp mailers)  
 "very.().:;<>[]\".VERY.\"very@\\\"very\".unusual"@strange.example.com (include non-letters character AND multiple at sign, the first one being double quoted)  
 user%example.com@example.org (% escaped mail route to user@example.com via example.org)  
 user-@example.org (local-part ending with non-alphanumeric character from the list of allowed printable characters)  
 postmaster@[123.123.123.123] (IP addresses are allowed instead of domains when in square brackets, but strongly discouraged)  
 postmaster@[IPv6:2001:0db8:85a3:0000:0000:8a2e:0370:7334] (IPv6 uses a different syntax)  
 Valid email addresses with SMTPUTF8  
 I❤CHOCOLATE🍫@example.com (emoji are only allowed with SMTPUTF8)

## Захватить CSV структуру

Создаются группы по значениям из CSV

(?:\s\*(?:\"([^\"]\*)\"|([^\,]+))\s\*,?)+?

REGULAR EXPRESSION 7 matches (85 steps, 0.1ms)

:/ (?:\s\*(?:\"([^\"]\*)\"|([^\,]+))\s\*,?)+? / gm

TEST STRING

Asset imported by script, asset.test.local, 192.168.230.222, aa:aa:aa:bb:bb:bb, "Vasya Petrov", Windows 10, 17993

## Захватить Syslog структуру

Разбираются сообщения по rfc <https://datatracker.ietf.org/doc/html/rfc5424> и

<https://datatracker.ietf.org/doc/html/rfc3164>

```
(?P<syslog>^<(P<priority>\d\d{2})|1[1-8]|\d19[01])>(P<version>\d{1,2})?s?(P<timestamp>(P<rfc3164>[A-Z][a-z][a-z]\s{1,2}\d{1,2}\s\d{2}[:]\d{2}[:]\d{2})|(P<rfc5424>\d{4}[-]\d{2}[-]\d{2}[T]\d{2}[:]\d{2}[:]\d{2}(?:\.\d{1,6})?(?:[+-]\d{2}[:]\d{2}[Z]))\s?(P<hostname>[\S]{1,255})\s?(P<appname>[\\w-]{1,48})?s?(P<procid>[\w]{1,128})?(P<delimMSG>[\:\s\|-\s\|:\s\|s](P<BIGmsg>(P<msgid>[\S]{1,32})?(P<delimMSGID>[\:\s\|-\s\|:\s\|s](P<structureddata>[.+[[]+|-]?(?:s(P<msg>.+))?)?)$)
```

REGULAR EXPRESSION 3 matches (291 steps, 0.2ms)

```
:/ (?P<syslog>^<(P<priority>\d\d{2})|1[1-8]|\d19[01])>(P<version>\d{1,2})?s?(P<timestamp>(P<rfc3164>[A-Z][a-z][a-z]\s{1,2}\d{1,2}\s\d{2}[:]\d{2}[:]\d{2})|(P<rfc5424>\d{4}[-]\d{2}[-]\d{2}[T]\d{2}[:]\d{2}[:]\d{2}(?:\.\d{1,6})?(?:[+-]\d{2}[:]\d{2}[Z]))\s?(P<hostname>[\S]{1,255})\s?(P<appname>[\\w-]{1,48})?s?(P<procid>[\w]{1,128})?(P<delimMSG>[\:\s\|-\s\|:\s\|s](P<BIGmsg>(P<msgid>[\S]{1,32})?(P<delimMSGID>[\:\s\|-\s\|:\s\|s](P<structureddata>[.+[[]+|-]?(?:s(P<msg>.+))?)?)$) / gm
```

TEST STRING

```
<34>Oct*11*22:14:15*mymachine*su:*'su*root'*failed*for*lonvick*on*/dev/pts/8
<165>1*2003-10-11T22:14:15.003Z*mymachine.example.com*evntslog*-ID47*[exampleSDID@32473*iut="3"*eventSource="Application"*eventID="1011"]*BOMAn*application*event*log*entry...
<0>Oct*22*10:52:12*scapegoat*1990*Oct*22*10:52:01*TZ-6*scapegoat.dmz.example.org*10.1.2.3*sched[0]:*That's*All*Folks!
```

## Захватить символы (не буквы и не цифры)

```
[^\w \xC0-\xFF]
```

REGULAR EXPRESSION 12 matches (24 steps, 0.1ms)

```
:/ [^\w \xC0-\xFF]
```

TEST STRING

```
Here is an example 15 docker-compose.yml which uses Docker's syslog log driver to forward NGINX docker container logs to Seq (i.e. whatever you see in stdout when you run docker logs -f <container-name>).
```

## Работа с многострочным сообщением



В regexp существуют следующие флаги:

- `i` - нечувствителен к регистру (по умолчанию false)
- `m` - многострочный режим, `^` и `$` соответствуют строке начала/конца в дополнение к тексту начала/конца (по умолчанию false)
- `s` - позволяет `.` (точке) совпадать с `\n` (по умолчанию false)
- `U` - не жадный режим, меняет местами значения `x*` и `x*?`, `x+` и `x+?` и т. д. (по умолчанию false)

Синтаксис флага: `хуз` (установить) или `-хуз` (очистить) или `ху-z` (установить `ху`, очистить `z`).

Устанавливаем флаг `s`

REGULAR EXPRESSION

1 match (9 steps, 0.0ms)

/ Рост: (?P<height>\d+)

/ gms

TEST STRING

Вес:100

Рост:77

Профессия:Инженер

Использование в KUMA:

Нормализация

☐ Использовать синтаксис CEF при нормализации

(?s)Рост:(?P<height>\d+)

Перенести названия полей в таблицу

+ Добавить регулярное выражение

Сопоставление

Исходные данные	Поле KUMA	Подпись	
height	FlexNumber1		

FlexNumber1

77

Type

Base

Исходное событие

Вес:100

Рост:77

Профессия:Инженер

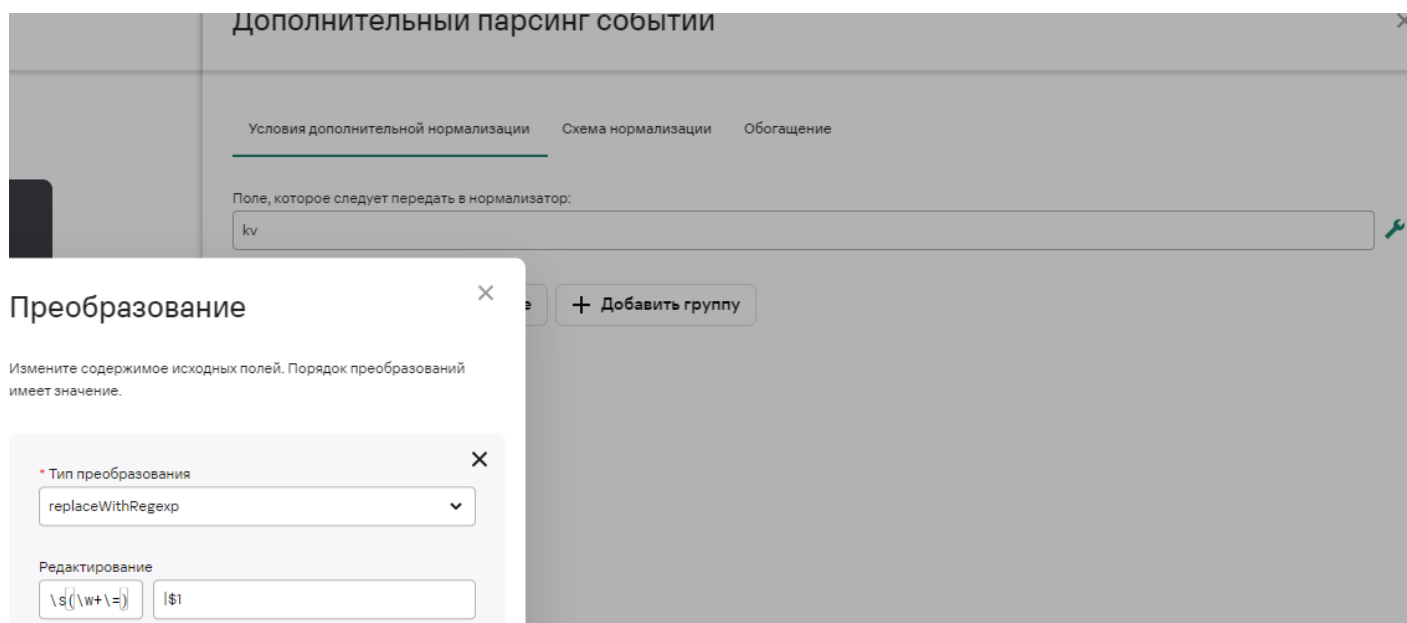
# Замена разделителя в структуре KV

Иногда в структуре KV разделитель пробел создает нам проблемы, пример такого события:

```
pid=\"29753\" appname=\"DBeaver 23.2.1 - SQLEditor <Script-40.sql>\" user=\"user2\" dbname=\"test_db\"  
rhost=\"comp.local(911)\" queryid=\"0\" command_tag=\"BIND\" sql_state=\"42501\"  
session_id=\"6662a086.7439\" session_seq=\"74\" session_start_time=\"2024-06-07 05:54:14 UTC\"  
virt_trans_id=\"5/231\" trans_id=\"0\" msg=STATEMENT: select lo_export(111,'tmp/pido')
```

В примере значение по ключу msg без кавычек, и оно будет некорректно парситься, поэтому можно, либо заменами добавить эти кавычки, либо воспользоваться приемом полегче и сделать разделитель, например |

Для этого используем функцию `replaceWithRegex`:



Вот как выглядит будет замена:

REGULAR EXPRESSION

13 matches (100 steps, 0.1ms)

/ \s(\w+=)

/ gm

TEST STRING

```
pid=\"29753\"|appname=\"DBeaver*23.2.1*-SQLEditor*<Script-40.sql>\"|user=\"user2\"|dbname=\"test_db\"|
rhost=\"comp.local(911)\"|queryid=\"0\"|command_tag=\"BIND\"|sql_state=\"42501\"|
session_id=\"6662a086.7439\"|session_seq=\"74\"|session_start_time=\"2024-06-07*05:54:14*UTC\"|
virt_trans_id=\"5/231\"|trans_id=\"0\"|msg=STATEMENT:*select*lo_export(111,'/tmp/pido')|
```

1:367

SUBSTITUTION

success (0.3ms)

|\\$1

```
pid=\"29753\"|appname=\"DBeaver*23.2.1*-SQLEditor*<Script-
40.sql>\"|user=\"user2\"|dbname=\"test_db\"|rhost=\"comp.local(911)\"|queryid=\"0\"|command_tag=\"BIND\
\"|sql_state=\"42501\"|session_id=\"6662a086.7439\"|session_seq=\"74\"|session_start_time=\"2024-06-07*
05:54:14*UTC\"|virt_trans_id=\"5/231\"|trans_id=\"0\"|msg=STATEMENT:*select*lo_export(111,'/tmp/pido')
```

Вот как это выглядит в KUMA:

Service	<a href="#">TEST BORIS (TCP/5577)</a>
Type	Base
Extra	<pre> appname: \DBeaver*23.2.1*-SQLEditor&lt;Script-40.sql&gt;\ command_tag: \BIND\ dbname: \test_db\ delim: * kv: pid=\29753\*appname=\DBeaver*23.2.1 *-SQLEditor&lt;Script-40.sql&gt;\*user=\user 2\*dbname=\test_db\*rhost=\comp.local (911)\*queryid=\0\*command_tag=\BIND \*sql_state=\42501\*session_id=\6662a0 86.7439\*session_seq=\74\*session_start _time=\2024-06-07*05:54:14*UTC\*virt_tra ns_id=\5/231\*trans_id=\0\*msg=STATEME NT:*select*lo_export(111,/tmp/pido) ms: .685 msg: STATEMENT:*select*lo_export(111,/tmp/ pido) pid: \29753\ queryid: \0\ rfc5424_modif: 2024-06-07*05:54:14.685 rhost: \comp.local(911)\ session_id: \6662a086.7439\ session_seq: \74\ session_start_time: \2024-06-07*05:54:14* UTC\ sql_state: \42501\ trans_id: \0\ user: \user2\ virt_trans_id: \5/231\ </pre>

Revision #11

Created 3 October 2023 13:16:14 by Boris RZR

Updated 23 July 2024 09:57:32 by Boris RZR