ManageEngine
DataSecurity Plus

How to write Regex to find sensitive data

www.datasecurityplus.com

Find sensitive personal data stored in your file servers using regular expressions or by matching keywords. Read on to learn what a regex is and how it can be used to find sensitive data.

What is a regex?

Regular expressions (regex) are strings of alphanumeric and special characters used to search for and find matching text patterns from log files, code, and other documents. In DataSecurity Plus, regex can be used to find sensitive personal data such as PCI, PII, or ePHI.

Example: If the email address "lisa@test.com" has to be found in a document, the regex would have to be written as following: \b[\w\-\.]+@([\w\-]+\.)+[\w\-]{2,4}\b

Note: Throughout this document, the regex are highlighted in green and the phrases for which the regex is written are highlighted in blue, for easier understanding.

How do you write a regex?

Writing a regex depends on the patterns you want to match for. If you want to match a specific sequence of literal characters like Data security, simply write Data security as your regex. If you want to match for a set of possible sequences, use a combination of alphanumeric and special characters, as described further in this document.

Any single literal character is a regex by itself. For example, if you want to find occurrences of the character d in the string "data security is a distinguished data risk assessment tool," just write d, and it will match four times. But it will only match three times for "Data security is a distinguished data risk assessment tool," since regex is case-sensitive.

There are 12 characters with special meaning in regular expressions. These special characters when used alone are construed as errors. When you need to use a special character as a part of your regex, it needs to be preceded with a '\'.
Example: Regex for Data.security is Data\.security

Special character	Meta characters	Intended meaning	Examples
Caret	^	It is the anchor for the character at the start of the string.	The regex ^d matches once in the string "dated", as it matches the character "d" at the beginning of the string.
Dollar sign	$	It is the anchor for the character at the end of the string.	The regex a$ matches once in the string "alpha", as it matches the character "a" at the end of the string.
Vertical bar	|	It is used to separate a set of alternatives.	The regex dat(a|e|o) will match data, date, or dato.
Period	.	It represents any single character.	The regex Dat. will match "Dat" followed by any character such as Data, Dat1, Dat-, etc.
Question mark	?	It represents the optional part of the string.	The regex Jan(uary)? will match both Jan and January
Asterisk	*	It matches zero or more of the characters preceding it.	The regex Dat*a will match both Datta and Daa
Plus	+	It matches one or more of the characters preceding it.	The regex Dat+a will match both Datta and Dattta but not Daa
Parenthesis	()	It groups characters in a regex. Its use can also be recalled later using shorthand, e.g., (\1).	The regex Dat(a|e) will match both Date and Data. The regex Jan(uary)[-]Febr(\1) will match January-February
Square brackets	[]	It represents any character inside. If ^ is used along with square brackets, it represents any character other than the one mentioned.	The regex Dat[a-f] will match "Dat" followed by any character in the range a-f, like Data, Datb, Datc, etc., but it will not match strings like Dato, Datu, Datn, etc. The regex Dat[^e] will match "Dat" followed by any character that is not "e", like Data, Dato, Dat1, Dat-, etc.
Curly braces	{}	It is used to quantify the range.	The regex Data{2,3} will match Dataa and Dataaa

Minus sign	-	It is used to indicate the range of characters allowed.	The reg Dat[a-z] will match "Dat" followed by any character in the range a-z, such as Data, Date, Datu, etc.
Backslash	\	It gives a special meaning to the character following it.	\n stands for new line; \w depicts a word of any size. The regex \w will match d, g, data, 12345, and more.

These 12 characters can be used in combination to find multiple variations of a specific phrase.

Example: Consider the regex \bInfo(rmation security|[-]?sec(urity|urities)?)\b. This regex can be used to find multiple variations of the phrase "Information security". It will match for the following variations: Information security, Infosec, Info-sec, Info-security, Info-securities, Infosecurity, and Infosecurities.

Explanation:
Info - Matches the string that starts with "Info"
(rmation security|[-]?sec(urity|urities)?) - The vertical bar gives two alternatives to match.

> **Alternative 1:** rmation security matches the string "rmation security". So, the variation "Information security" will be matched.
>
> **Alternative 2:** [-]?sec matches the character "-" optionally, followed by the string sec. So, the variations Info-sec and Infosec will be matched. (urity|urities)? matches the strings urity or urities optionally. So, the variations Info-security, Info-securities, Infosecurity, and Infosecurities will be matched.

In addition to the above-described characters, there are also a few other special meta characters that make the process of constructing regular expressions easier and faster. Given below is the list of special meta characters commonly used:

Special meta-character	Intended meaning	Examples
\d	It denotes any whole number (0-9).	The regex \d\d will match two consecutive whole numbers such as 79 and 30 but not 5.
\w	It denotes any word character (alphanumeric or underscore).	The regex \w\w will match two consecutive alphanumeric or underscores such as do, d7, _c, etc., but not 8.

\w	It denotes any symbol that is not alphanumeric or underscore.	The regex \W\W will match two consecutive symbols such as %% and #&.
\b	It denotes a word boundary position at the beginning or end of a string.	The regex \bdeer\b will match the string deer.
{n}	It denotes the number of times the preceding character is to be repeated.	The regex Dat{3} will only match Dattt, as the character "t" is repeated thrice. The regex will not match for strings like Datt, Datttt, Dat, etc.
{n,m}	Within a set of brackets, n denotes the least number and m denotes the maximum number of times the preceding character is to be repeated.	The regex Data{3,6} will match Dataaa and Dataaaa but not Dataa.
\s	It denotes any white space character including 'space' and 'tab'	The regex Data\ssecurity will match Data<space>security and Data<tab>security
\S	It denotes any non-white space character.	The regex \S\S will match two consecutive non-white space characters, such as &H and t8 but not 5.
\D	It denotes any non-digit character.	The regex \D\D will match two consecutive non-digit characters such as to and t# but not g7.

Let's consider a few examples of personal data regexes and decode it.

American Express card numbers:
RegEx: \b3[4,7][0,9]{13}\b
3 - The string starts with 3
[4,7] - Followed by either 4 or 7
[0,9]{13} - Followed by 13 characters within the range 0-9
Examples: 372418640982660 and 345613290542766

The regex is surrounded by \b on both ends to make sure it doesn't match for the same pattern within a longer string like 103923724186409826601393.

Email Address:

RegEx: \b[\w\-\.]+@([\w\-]+\.)+[\w\-]{2,4}\b

[\w\-\.] - The string starts with one or more characters that is alphanumeric, underscore, "-", or "."

+@ - Followed by the character "@"

([\w\-] - Followed by one or more characters that is alphanumeric or underscore

+\.] - Followed by the character "."

+[\w\-]{2,4} - Followed by 2 to 4 characters that is alphanumeric, "_", or "-"

Examples: dave.cs@xyz.com and robert52@dst2.in

Social Security Number (SSN)

RegEx: \b\d{3}\-\d{2}\-\d{4}\b

\d{3} - The string starts with three whole numbers

\- - Followed by the character "-"

\d{2} - Followed by two whole numbers

\- - Followed by the character "-"

\d{4} - Followed by the four whole numbers

Examples: 123-45-6789 and 353-09-0007

The regex is surrounded by \b on both ends to make sure it doesn't match for the same pattern within a longer string like 999123-12-1234999999999.

Note: This document is intended to give the users an introduction to constructing regular expressions. In case you require any additional assistance, feel free to raise a request with our support team, and we will get back to you immediately.

Additional resources

- Regexr is an online tool to learn, build, and test regular expressions. It also offers a searchable database of patterns submitted by online peer community.
- Regexlib is a library of most commonly used regular expressions.
- Regular-expressions.info provides a wide range of in-depth information on how-to construct a regular expression.
- Regex101 is an online regular expressions tester and debugger.

DataSecurity Plus

ManageEngine DataSecurity Plus is a unified data visibility and security platform. It audits file changes in real time, triggers instant responses to critical events, shuts down ransomware intrusions, and helps organizations comply with numerous IT regulations. It analyzes file storage and security permissions, deletes junk files, and detects security vulnerabilities. Users can assess the risks associated with sensitive data storage by locating and classifying files containing personally identifiable information (PII), payment card information (PCI), and electronic protected health information (ePHI). It also prevents data leaks via USBs, email, printers, and web applications; monitors file integrity; and audits cloud application usage. Together, these capabilities ensure the all-round protection of data at rest, data in use, and data in motion.

To explore these features and see DataSecurity Plus in action, check out the online demo.
To learn more about DataSecurity Plus, visit www.datasecurityplus.com.

Download free trial | $ Get a quote

Explore DataSecurity Plus' capabilities

File server auditing

Audit and report on file accesses and modifications, with real-time alerts and automated responses for critical file activities.

Learn more

File analysis

Analyze file security and storage, manage junk files, optimize disk space usage, and identify permission vulnerabilities.

Learn more

Data risk assessment

Discover and classify files containing sensitive data such as PII, PCI, and ePHI by combining content inspection and contextual analysis.

Learn more

Data leak prevention

Detect and disrupt data leaks via USBs, email, web applications, and printers; monitor endpoint file activity; and more.

Learn more

Cloud protection

Track enterprise web traffic and enforce policies to block the use of inappropriate, risky, or malicious web applications.

Learn more

Our Products

AD360 | Log360 | ADAudit Plus | EventLog Analyzer

Exchange Reporter Plus | SharePoint Manager Plus