Regular Expressions

Regular expressions (`RegEx`) are an art of expression language to search for patterns in text and files. They can be used to find and replace text, analyze data, validate input, perform searches, and more. In simple terms, they are a filter criterion that can be used to analyze and manipulate strings. They are available in various programming languages and programs and are used in many different ways and functions.

A regular expression is a sequence of letters and symbols that form a search pattern. In addition, regular expressions can be created with patterns called metacharacters. Meta characters are symbols that define the search pattern but have no literal meaning. We can use it in tools like `grep` or `sed` or others. Often regex is implemented in web applications for the validation of user input.

Grouping

Among other things, regex offers us the possibility to group the desired search patterns. Basically, regex follows three different concepts, which are distinguished by the three different brackets:

Grouping Operators

	Operators	Description
1	`(a)`	The round brackets are used to group parts of a regex. Within the brackets, you can define further patterns which should be processed together.
2	`[a-z]`	The square brackets are used to define character classes. Inside the brackets, you can specify a list of characters to search for.
3	`{1,10}`	The curly brackets are used to define quantifiers. Inside the brackets, you can specify a number or a range that indicates how often a previous pattern should be repeated.
4	`|`	Also called the OR operator and shows results when one of the two expressions matches
5	`.*`	Also called the AND operator and displayed results only if both expressions match

Suppose we use the `OR` operator. The regex searches for one of the given search parameters. In the next example, we search for lines containing the word `my` or `false`. To use these operators, you need to apply the extended regex using the `-E` option in grep.

OR operator

OR operator

```
cry0l1t3@htb:~$ grep -E "(my|false)" /etc/passwd

lxd:x:105:65534::/var/lib/lxd/:/bin/false
pollinate:x:109:1::/var/cache/pollinate:/bin/false
mysql:x:116:120:MySQL Server,,:/nonexistent:/bin/false
```

Since one of the two search parameters always occurs in the three lines, all three lines are displayed accordingly. However, if we use the `AND` operator, we will get a different result for the same search parameters.

AND operator

AND operator

```
cry0l1t3@htb:~$ grep -E "(my.*false)" /etc/passwd

mysql:x:116:120:MySQL Server,,:/nonexistent:/bin/false
```

Basically, what we are saying with this command is that we are looking for a line where we want to see both `my` and `false`. A simplified example would also be to use `grep` twice and look like this:

AND operator

```
cry0l1t3@htb:~$ grep -E "my" /etc/passwd | grep -E "false"

mysql:x:116:120:MySQL Server,,:/nonexistent:/bin/false
```

Here are some optional tasks to practice regex that can help us to handle it better and more efficiently. For all exercises, we will use the `/etc/ssh/sshd_config` file on our `Pwnbox` instance.

1	Show all lines that do not contain the `#` character.
2	Search for all lines that contain a word that starts with `Permit`.
3	Search for all lines that contain a word ending with `Authentication`.
4	Search for all lines containing the word `Key`.
5	Search for all lines beginning with `Password` and containing `yes`.
6	Search for all lines that end with `yes`.

Start Instance

0 / 1 spawns left

Waiting to start...