



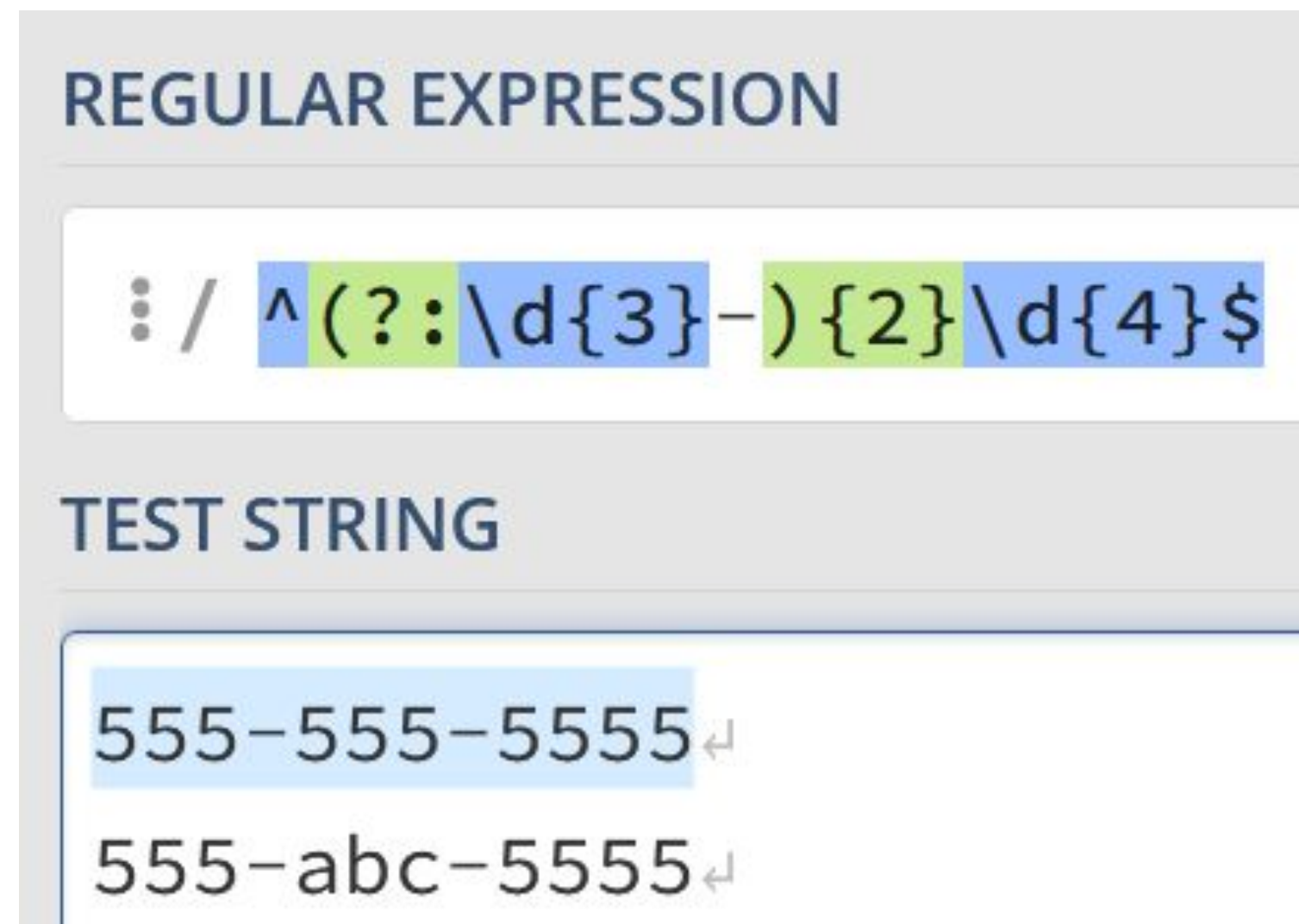
REGEX

Expresiones regulares

- Una **expresión regular**, comúnmente conocida como **regex** (del inglés "**regular expression**"), es una secuencia de caracteres que define un patrón de búsqueda.
- Este patrón se utiliza para encontrar coincidencias dentro de cadenas de texto, permitiendo identificar, extraer, reemplazar o manipular texto de manera eficiente.
- Por ejemplo, si quieres encontrar todas las direcciones de correo electrónico en un documento, podrías usar una expresión regular para buscar patrones que coincidan con la estructura típica de un correo electrónico (como ``nombre@dominio.com``).
- Las expresiones regulares son muy útiles en programación y análisis de datos, ya que permiten realizar búsquedas avanzadas y manipulaciones de texto con gran precisión.
- Existen multitud de páginas web y software libre donde poder realizar pruebas con expresiones regulares sobre logs específicos y obtener la expresión que "cubra" todo el contenido del log para poder extraer de él los datos.
- Referencias a algunas webs:
 - www.regexr.com
 - www.regex101.com
 - www.regexpal.com

Aplicaciones Principales del REGEX

- Algunas de las principales aplicaciones de las expresiones regulares son:
 - **Validación de datos:** Las expresiones regulares son muy útiles para validar y verificar que una cadena de texto cumpla con un formato específico. Por ejemplo, podemos utilizar una expresión regular para validar que un número de teléfono esté en el formato correcto.
 - **Búsqueda y extracción de información:** Las expresiones regulares nos permiten buscar y extraer información específica de un texto. Por ejemplo, podemos utilizar una expresión regular para extraer todos los números de teléfono de un documento.
 - **Reemplazo de texto:** Las expresiones regulares también nos permiten reemplazar texto en base a un patrón específico. Por ejemplo, podemos utilizar una expresión regular para reemplazar todas las ocurrencias de una palabra por otra en un documento.
 - **Procesamiento de texto en lenguajes de programación:** Las expresiones regulares son ampliamente utilizadas en lenguajes de programación para manipular y procesar cadenas de texto de manera eficiente.
- Son muy útiles y versátiles, ya que nos permite buscar, validar y manipular patrones en cadenas de texto de manera eficiente y precisa.



<https://custommapposter.com/article/the-complete-guide-to-regular-expressions-gex-coderpad/806>

Principales Uso del REGEX

- Algunos de los principales usos de las expresiones regulares son:
 - **Validación de formularios:** Las expresiones regulares son muy útiles para verificar si los datos ingresados en un formulario cumplen con un formato específico, como, por ejemplo, una dirección de correo electrónico válida o un número de teléfono.
 - **Análisis de logs:** En el análisis de registros o logs, las expresiones regulares se utilizan para extraer información específica de los registros, como fechas, direcciones IP o mensajes de error.
 - **Manipulación de texto:** Con las expresiones regulares se pueden realizar operaciones como sustituir palabras o caracteres en un texto, eliminar espacios o caracteres no deseados, o dividir una cadena en partes.
 - **Extracción de datos:** Si se tiene un texto con información estructurada, como una lista de nombres o direcciones, las expresiones regulares permiten extraer de manera precisa los datos deseados.

REGULAR EXPRESSION

```
!r" gs://(.*)/(.*)
```

TEST STRING

```
gs://bucket-name/filepath.txt↵
```

```
gs://bucket-name/folder/file.txt↵
```

https://engineeringfordatascience.com/posts/how_to_extract_bucket_and_filename_info_from_gcs_uri/

Expresiones regulares

- REGEX Básico

Carácter	Significado
^	Indica el principio de una cadena
\$	Indica el final de una cadena
()	Un agrupamiento de parte de una expresión
[]	Un conjunto de caracteres dentro de la expresión Para indicar que solo podrá tener los caracteres a, b, c y d, así como también números del 1 al 7: [a-d1-7]
{}	indica un número o intervalo de longitud de la expresión Para indicar que serán 2 números utilizamos: \d{2}
.	Cualquier carácter, salvo el salto de línea
?	0-1 ocurrencias de la expresión
+	1-n ocurrencias de la expresión
*	0-n ocurrencias de la expresión

Expresiones regulares

- REGEX Básico

Carácter	Significado
\	Para escribir un carácter especial como los anteriores y que sea tratado como un literal Si siempre se recibe un carácter especial como el punto "." usamos \.
 	Para indicar una disyunción lógica (para elegir entre dos valores: a b se tiene que cumplir al menos uno de los dos) Solo acepta el valor de YES o NO usamos YES NO.
\s	Ocurrencia de espacios, tabs...
\d	Ocurrencia de dígitos
\w	Coincide con cualquier carácter alfanumérico
\W	negado de \w
\D	negado de \d
\S	negado de \s

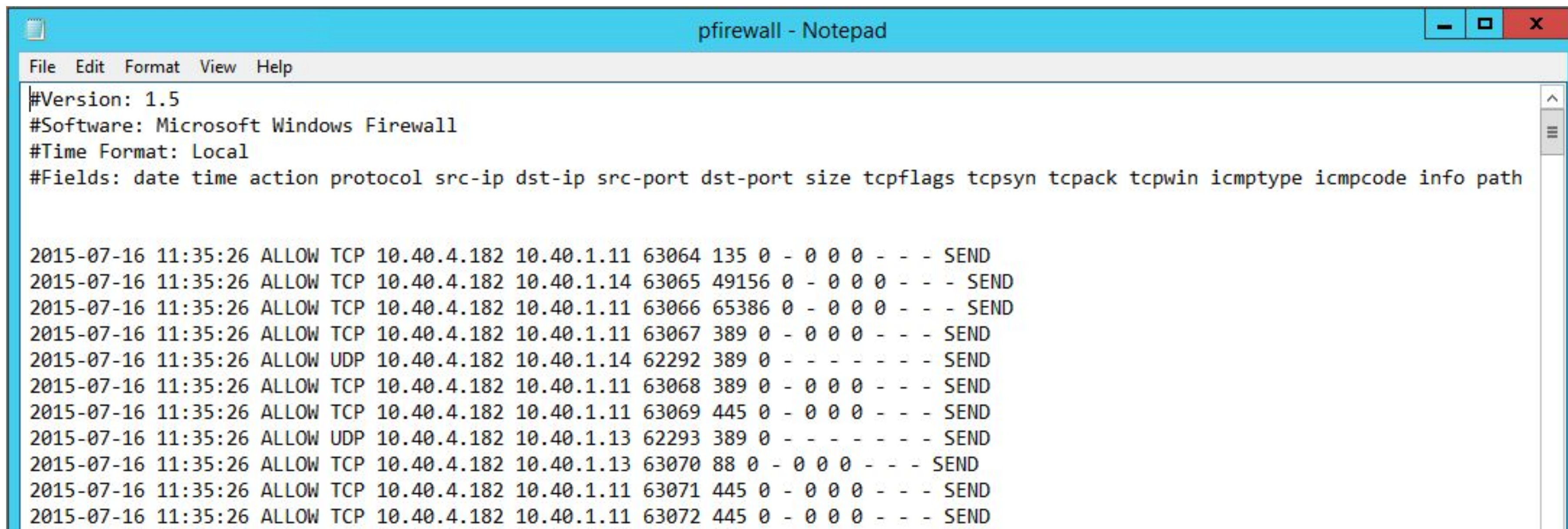
Acciones realizadas por los recolectores

- Tanto para el modo de recolección activa como la pasiva, los sistemas recolectores o los agentes/conectores, realizan las siguientes acciones sobre los logs:



Parseo de Logs

- Se entiende por parseo de logs a la separación en campos de los distintos elementos de un log. Dicha separación en campos se suele realizar mediante expresiones regulares.
- Ejemplo de log de un firewall de Windows:



```
pfirewall - Notepad
File Edit Format View Help
#Version: 1.5
#Software: Microsoft Windows Firewall
#Time Format: Local
#Fields: date time action protocol src-ip dst-ip src-port dst-port size tcpflags tcpsyn tcppack tcpwin icmptype icmpcode info path

2015-07-16 11:35:26 ALLOW TCP 10.40.4.182 10.40.1.11 63064 135 0 - 0 0 0 - - - SEND
2015-07-16 11:35:26 ALLOW TCP 10.40.4.182 10.40.1.14 63065 49156 0 - 0 0 0 - - - SEND
2015-07-16 11:35:26 ALLOW TCP 10.40.4.182 10.40.1.11 63066 65386 0 - 0 0 0 - - - SEND
2015-07-16 11:35:26 ALLOW TCP 10.40.4.182 10.40.1.11 63067 389 0 - 0 0 0 - - - SEND
2015-07-16 11:35:26 ALLOW UDP 10.40.4.182 10.40.1.14 62292 389 0 - - - - - - - SEND
2015-07-16 11:35:26 ALLOW TCP 10.40.4.182 10.40.1.11 63068 389 0 - 0 0 0 - - - SEND
2015-07-16 11:35:26 ALLOW TCP 10.40.4.182 10.40.1.11 63069 445 0 - 0 0 0 - - - SEND
2015-07-16 11:35:26 ALLOW UDP 10.40.4.182 10.40.1.13 62293 389 0 - - - - - - - SEND
2015-07-16 11:35:26 ALLOW TCP 10.40.4.182 10.40.1.13 63070 88 0 - 0 0 0 - - - SEND
2015-07-16 11:35:26 ALLOW TCP 10.40.4.182 10.40.1.11 63071 445 0 - 0 0 0 - - - SEND
2015-07-16 11:35:26 ALLOW TCP 10.40.4.182 10.40.1.11 63072 445 0 - 0 0 0 - - - SEND
```


Expresiones regulares

- **Cómo afrontar el parseo de un evento/log de forma efectiva.**
 - El primer paso será determinar cuál es el separador de campos.
 - Hacer un mapeo de campos.
 - Para este paso debemos apoyarnos en la documentación oficial del producto en cuestión para garantizar la correcta elección de cada uno de los campos.
 - Realizar la expresión regular para este tipo de evento.
- ***OJO!!! No siempre nos puede interesar mapear todos los campos de un log, ya que algunos campos pueden aportarnos poca o nula información útil.**

Parseo de Logs

- Tomando la primera línea como ejemplo, podemos ver la siguiente información:

- **2015-07-16 11:35:26 ALLOW TCP 10.40.4.182 10.40.1.11 63064 135 0 – 0 0 0 - - - SEND**

- Al ver esta información podemos interpretar:

- **Fecha:** 2015-07-16
 - **Hora:** 11:35:26
 - **Acción realizada:** ALLOW
 - **Protocolo:** TCP
 - **Dirección IP origen:** 10.40.4.182
 - **Dirección IP destino:** 10.40.1.11
 - **Puerto Origen:** 63064
 - **Puerto Destino:** 135
 - **Tamaño:** 0
 - **Tcpflags:** -
 - **Tcpsyn:** 0
 - **Tcpack:** 0
 - **Tcpwin:** 0
 - **Icmptype:** -
 - **Icmpmode:** -
 - **Info:** -
 - **Path:** SEND

- El **recolector/conector/agente**, sin saber lo que es una fecha, una hora o una acción, puede, mediante expresiones regulares, asociar que lo que aparezca en los primeros 10 caracteres del evento es una fecha, que después aparece un espacio, luego los siguientes 8 caracteres representan la hora, luego aparece otro espacio...tras el espacio, la siguiente secuencia de letras representa la acción, etc.....

Expresiones regulares

- **Ejemplos reales**

- Open VPN Client:
 - <https://regex101.com/r/1E4RrE/2>
- LOGRHYTHM:
 - <https://regex101.com/r/nW9nC2/1>
- FW Log:
 - <https://regex101.com/r/wp2oYf/1>
- Mikrotik Firewall
 - <https://regex101.com/r/sW8kS0/1>
- command df -T on Linux
 - <https://regex101.com/r/IQXeFI/4>
- Windows File Name
 - <https://regex101.com/r/VhOqt3/1>
- Java error
 - <https://regex101.com/r/iX4vW9/1>

