Configuración y validación de expresiones regulares en Cisco ESA y CES

Contenido

- Introducción
- Antecedentes
- Diccionarios y términos de búsqueda
  - Ejemplos de caracteres especiales y su sintaxis de escape
- Limitación del uso de expresiones regulares
- Filtros de mensajes, filtros de contenido y diccionarios
- Motor de expresiones regulares
- Caracteres no ASCII y límites de palabras
- Escribir filtros eficientes
- PDF y expresiones regulares
- Prueba de expresiones regulares
- Introducción de la expresión en un filtro de contenido y en un diccionario
  - Introducción de la expresión en un filtro de contenido
  - Introducción de la expresión en un diccionario
    - Acerca de "Coincidir palabras enteras"
- Clasificación de costes Regex en Cisco ESA
  - Patrones de alto riesgo más caros
    - Cuantificadores anidados (peor caso)
    - Codicioso.* Seguido de un patrón obligatorio
    - Alternaciones grandes con prefijos compartidos
    - Coste medio: utilice con precaución
    - Cuantificadores perezosos (+?, *?)
    - Clases de caracteres muy genéricos
  - Bajo coste: patrones seguros y eficientes
    - Literales fijos
    - Utilizar delimitadores para limitar el ámbito de búsqueda
    - Clases de caracteres específicas
    - Patrones estructurados y restringidos
  - Guía práctica para Cisco ESA
  - Comparación de rendimiento de Regex (contexto de Cisco ESA)
- Conclusión
- Documentación

Introducción

Este documento describe cómo ESA y CES utilizan expresiones regulares en los filtros, las diferencias de comportamiento clave y la necesidad de realizar pruebas antes de la aplicación.

Antecedentes

Este documento describe cómo Cisco Email Security Appliance (ESA) y Cisco Cloud Email Security (CES) gestionan las expresiones regulares cuando se utilizan dentro de los filtros de mensajes y los filtros de contenido. Se centra específicamente en comprender cómo se comportan las expresiones regulares en estos componentes y cómo interactúan con los encabezados de correo electrónico, el contenido principal y los archivos adjuntos.

Es importante aclarar desde el principio que el motor de expresiones regulares utilizado en el módulo DLP se comporta de manera diferente. Por lo tanto, todo lo descrito en este documento se aplica exclusivamente a los filtros de mensajes y los filtros de contenido y no se aplica a las políticas de DLP.

Al trabajar con expresiones regulares en ESA, los administradores deben comprender que el contenido de correo electrónico no se evalúa de la misma manera que se muestra visualmente en un cliente de correo. Los mensajes de correo electrónico contienen información del sobre, encabezados estructurados, partes MIME y contenido potencialmente codificado. Como resultado, las comparaciones realizadas por los filtros pueden producir resultados inesperados si la estructura del mensaje y el comportamiento de regex no se entienden completamente.

Por este motivo, cualquier filtro nuevo que utilice expresiones regulares siempre se puede habilitar en el modo de supervisión antes de la aplicación. Esto permite la validación frente al tráfico real y evita el bloqueo involuntario o el impacto en el rendimiento.

Diccionarios y términos de búsqueda

Al crear un filtro de mensajes o un filtro de contenido, el término introducido en muchas condiciones se interpreta como una expresión regular. Se trata de un concepto fundamental: incluso cuando el administrador intenta hacer coincidir el texto literal, el ESA puede procesar la entrada mediante la lógica regex.

Esto no se aplica uniformemente a todos los tipos de condiciones. Por ejemplo, cuando se busca una dirección IP específica en determinadas condiciones estructuradas, el valor no se interpreta como una expresión regular. Sin embargo, al buscar dentro del encabezado Asunto, el cuerpo del mensaje, un campo de encabezado específico o un nombre de archivo de datos adjuntos, el valor se suele tratar como un patrón de expresiones regulares.

Un ejemplo común ilustra esto claramente. Supongamos que el objetivo es bloquear los correos electrónicos con el asunto:

```
Receipt number (123456)
```

Dado que los paréntesis son caracteres especiales en las expresiones regulares (que se utilizan para agrupar), se deben incluir caracteres de escape.

La expresión correcta sería:

```
Receipt number \(123456\)
```

Si los paréntesis no son de escape, el motor regex los interpreta como operadores de agrupamiento en lugar de caracteres literales. Dependiendo del patrón, esto puede causar coincidencias no intencionadas o un comportamiento diferente al esperado.

Debido a esto, es esencial entender qué caracteres tienen un significado especial en regex y asegurarse de que se escapan correctamente cuando se requiere la coincidencia literal.

Ejemplos de caracteres especiales y su sintaxis de escape

La primera columna muestra un texto de ejemplo que contiene caracteres especiales y la segunda columna muestra cómo se debe escribir la sintaxis de expresión regular correcta para que coincida con ese texto literal en Cisco ESA (regex estilo Python).

Texto literal que debe coincidir	Sintaxis de expresión regular correcta
Número de recibo (123456)	Número de recibo \(123456\)
user@example.com	user@example\.com
www.test.abc	www\.test\.abc
file_name.txt	file_name\.txt
el precio es 10,50	el precio es 10\,50
C:\Users\Admin	C:\\Users\\Admin
[CONFIDENCIAL]	\[CONFIDENCIAL\]
{Invoice}	\{factura\}
+34 600 123 456	\+34 600 123 456
pregunta?	pregunta\?
100% garantizado	100% garantizado (% no requiere escape)
asterisco * símbolo	símbolo de asterisco *
A|B	A|B
intercalar ^inicio	símbolo de intercalación \^start
$100	dólar \$100

Limitación del uso de expresiones regulares

Las expresiones regulares deben utilizarse con cuidado y sólo cuando sea necesario. Aunque proporcionan potentes funciones de coincidencia, las expresiones excesivas o mal diseñadas pueden aumentar el tiempo de procesamiento de mensajes y producir coincidencias no deseadas.

Una construcción particular que requiere precaución es .* , que representa "cualquier carácter, cero o más veces". Cuando se coloca al principio o al final de una expresión, puede provocar un seguimiento excesivo y una sobrecarga de procesamiento innecesaria.

La documentación de Cisco indica que las entradas que utilizan .* al principio o al final pueden hacer que el sistema se bloquee bajo ciertas condiciones cuando coincida con partes MIME específicas. Por este motivo, Cisco recomienda evitar el uso de los comandos de inicio o final .* siempre que sea posible.

En muchos escenarios, los administradores utilizan patrones como .*Invoice.*cuando podían simplemente escribir factura y producir el mismo resultado práctico en ESA. Dado que el motor de exploración ya busca en las áreas de contenido relevantes, rodear una palabra con .* es a menudo redundante y computacionalmente ineficiente.

⚠ Precaución: La recomendación general es mantener las expresiones regulares lo más simples y precisas posible.

Filtros de mensajes, filtros de contenido y diccionarios

Cisco ESA proporciona varios mecanismos para evaluar mensajes y aplicar acciones. Los filtros de mensajes actúan al principio de la canalización y utilizan una sintaxis de estilo de secuencias de comandos. Son extremadamente flexibles y permiten una lógica avanzada que incluye datos de sobre, encabezados y propiedades de adjuntos. Sin embargo, debido a que se ejecutan en las fases tempranas de la cadena de procesamiento, los filtros de mensajes ineficientes pueden afectar negativamente al rendimiento.

Los filtros de contenido se configuran a través de la interfaz gráfica y funcionan después de que se haya aceptado el mensaje. En la mayoría de los casos prácticos de inspección de contenido, los filtros de contenido son más fáciles de gestionar y más seguros desde el punto de vista del rendimiento.

Tanto en los filtros de mensajes como en los filtros de contenido, las expresiones regulares se pueden introducir directamente en una condición o indirectamente mediante el uso de diccionarios.

Los diccionarios permiten a los administradores centralizar términos de búsqueda reutilizables. Cada entrada está escrita en una línea independiente y puede ser texto sin formato o una expresión regular. Los diccionarios también admiten caracteres no ASCII, lo que los hace

adecuados para entornos multilingües.

En algunas situaciones, ciertas construcciones complejas de expresiones regulares no pueden comportarse de manera idéntica dentro de los diccionarios. Cuando esto ocurre, la expresión regular se debe colocar directamente en la condición de filtro en lugar de dentro del diccionario.

Cisco ESA permite la creación de hasta 150 diccionarios de contenido. De forma predeterminada, se pueden configurar 100 diccionarios a menos que se modifique el límite a través de la CLI mediante el comando dictionaryconfig.

Los diccionarios también pueden implementar la ponderación de términos. A cada término se le puede asignar una ponderación numérica y, cuando el ESA analiza un mensaje, multiplica el número de apariciones de ese término por su ponderación. La puntuación resultante se compara con un umbral definido en el filtro. Este modelo de puntuación permite una aplicación de políticas más flexible y gradual.

Además, los diccionarios pueden incluir identificadores inteligentes, que son detectores algorítmicos de patrones numéricos estructurados, como números de la seguridad social o identificadores bancarios.

Motor de expresiones regulares

Cisco ESA utiliza expresiones regulares basadas en el estilo de módulo de Python re. Aunque esto proporciona compatibilidad con la sintaxis común de Python regex, no todas las funciones avanzadas admitidas en entornos Python completos son necesariamente compatibles con ESA.

Para la coincidencia exacta de cadenas, las expresiones se deben anclar utilizando ^ al principio y $ al final. Sin estos anclajes, el motor regex puede hacer coincidir subcadenas en lugar de valores completos.

Por ejemplo, la expresión:

```
sun.com
```

Coincidir con cadenas como:

```
thegodsunocommando
```

Sin embargo, la expresión:

```
^sun\.com$
```

Coincida sólo con la cadena exacta sun.com.

Cuando se hace coincidir una cadena vacía, es importante no utilizar "", ya que esto efectivamente coincide con todas las cadenas. En su lugar, la expresión correcta es:

```
^$
```

Dado que Cisco ESA utiliza expresiones regulares de estilo Python, hay un par de maneras de realizar una comparación que no discrimine mayúsculas y minúsculas.

De forma predeterminada, como se ha mencionado, las expresiones regulares distinguen entre mayúsculas y minúsculas. Esto significa buscar:

```
foo
```

Solo coincide con foo, pero no con FOO, Foo o Foo.

Si desea realizar una coincidencia que no distinga entre mayúsculas y minúsculas, puede utilizar el marcador en línea (?i) al principio de la expresión regular. Esto le dice al motor regex que ignore las mayúsculas y minúsculas para el resto del patrón.

Por ejemplo:

```
(?i)foo
```

Coincidencia de esta expresión:

- foo
- FOO
- Foo
- OfO

Si desea hacer coincidir toda la cadena exactamente, ignorando las mayúsculas y minúsculas, puede combinar el indicador que no distingue entre mayúsculas y minúsculas con los delimitadores:

```
(?i)^foo$
```

Esto garantiza que el valor total sea exactamente "foo", independientemente de la capitalización.

Otra alternativa (menos práctica) sería definir explícitamente todas las combinaciones posibles utilizando clases de caracteres, por ejemplo:

```
[Ff][Oo][Oo]
```

Sin embargo, este enfoque resulta difícil de mantener y no se recomienda cuando se puede utilizar el indicador (?i) en su lugar.

En la mayoría de los escenarios ESA, el método preferido y más limpio para la coincidencia sin distinción de mayúsculas y minúsculas es utilizar:

```
(?i)
```

al principio de la expresión regular.

Caracteres no ASCII y límites de palabras

En los lenguajes que utilizan conjuntos de caracteres de doble byte, los conceptos de límites de palabras o de mayúsculas y minúsculas no pueden comportarse como se espera. Las expresiones complejas que dependen de construcciones como \w pueden producir resultados incoherentes cuando se desconoce la codificación o la configuración regional.

En tales casos, puede ser aconsejable inhabilitar la aplicación de límites de palabras en la configuración del diccionario o simplificar la expresión para evitar la dependencia de clases de caracteres ambiguas.

Cuando se trabaja con diccionarios que no son ASCII, la visualización CLI no puede representar los caracteres correctamente según la codificación de terminal. En esos casos, el enfoque recomendado es exportar el diccionario a un archivo de texto, editarlo externamente y volver a importarlo.

Escribir filtros eficientes

La eficacia es fundamental a la hora de escribir filtros, especialmente en entornos de gran volumen. Un error común es escribir largas cadenas de condiciones OR para coincidencias similares.

Por ejemplo, si se comprueban docenas de extensiones de adjuntos individualmente, el motor regex se inicializa repetidamente. Esto aumenta el uso de la CPU y reduce el mantenimiento.

En lugar de escribir muchas comparaciones separadas, agruparlas usando la alternancia dentro de una sola expresión regular reduce significativamente la sobrecarga de procesamiento. Esto reduce el número de veces que se invoca el motor regex y facilita el mantenimiento del filtro.

Un diseño eficiente del filtro no solo tiene que ver con la legibilidad, sino que afecta directamente al rendimiento del sistema.

PDF y expresiones regulares

La coincidencia de contenido dentro de archivos PDF puede producir resultados inesperados dependiendo de cómo se generó el PDF. Algunos PDF no contienen espacios lógicos ni saltos de línea en su representación interna. El motor de exploración intenta reconstruir el espaciado lógico basándose en el posicionamiento de las palabras.

Si una palabra se construye utilizando varias fuentes o tamaños de fuente, la representación interna puede fragmentar el texto. Por ejemplo, la palabra "llamada" puede interpretarse internamente como "llamada" o "c a l lout".

En estos casos, puede fallar el intento de coincidencia con la expresión "llamada" porque la representación interna no contiene esa cadena contigua exacta. Los administradores deben ser conscientes de esta limitación al diseñar políticas basadas en contenido dirigidas a los archivos adjuntos PDF.

Prueba de expresiones regulares

Probar las expresiones regulares antes de implementarlas en producción es un requisito operativo crítico. Una expresión regular que parece sintácticamente correcta puede comportarse de manera muy diferente cuando se evalúa frente al tráfico de correo electrónico real. Sin las pruebas adecuadas, un filtro puede generar falsos positivos, no detectar los patrones esperados, introducir una sobrecarga de rendimiento o interrumpir involuntariamente el flujo de correo legítimo.

El ensayo debe abordarse como un proceso estructurado en dos fases para minimizar el riesgo antes de activar un filtro en producción.

Fase 1 - Diseño y validación de expresiones regulares
La primera fase se centra en diseñar y validar la propia expresión regular antes de integrarla en Cisco ESA.

1. Uso de regex101 o herramientas similares

Las plataformas en línea como http://regex101.com (o herramientas equivalentes) son muy útiles durante la fase de diseño. Al utilizar estas herramientas, debe seleccionarse el sabor de Python para aproximarse al motor de expresiones regulares de ESA.

Estas plataformas permiten a los administradores:

- Validar la corrección de sintaxis

- Confirme que los caracteres especiales se han escapado correctamente
- Probar tanto casos coincidentes como no coincidentes
- Visualizar el comportamiento de agrupamiento y cuantificador
- Identificar construcciones potencialmente codiciosas como .*

Sin embargo, estas herramientas simulan el comportamiento de expresiones regulares de Python estándar y pueden admitir funciones que no están completamente implementadas en Cisco ESA. Por lo tanto, deben considerarse herramientas de validación preliminares en lugar de pruebas de compatibilidad definitivas.

2. Uso de modelos de IA (ChatGPT, Copilot, ...)

Las secretarias basadas en IA pueden acelerar la creación de expresiones regulares, especialmente en situaciones de coincidencia complejas. Al describir el comportamiento deseado en lenguaje natural, los administradores pueden obtener una propuesta de regex inicial que luego se puede refinar.

Las herramientas de IA son especialmente útiles para:

- Generación de expresiones agrupadas complejas
- Conversión de los requisitos empresariales en sintaxis regex
- Simplificación de condiciones largas basadas en OR en alternancias agrupadas

Sin embargo, las expresiones generadas por IA siempre deben ser revisadas críticamente. Pueden introducir ineficiencias, construcciones no admitidas o lógica demasiado compleja. La asistencia de Amnistía Internacional debe tratarse como ayuda de redacción, no como validación final. Cada expresión generada por IA debe ser probada usando métodos de validación estructurados.

Fase 2: Validación del comportamiento de los filtros en Cisco ESA

Una vez validada la expresión en sí, la segunda fase se centra en confirmar cómo se comporta dentro de Cisco ESA cuando se aplica al procesamiento de mensajes reales.

1. Uso de la Función Trace en la Consola CES

La función Seguimiento de la consola de Cisco Email Security (CES) permite a los administradores simular y analizar cómo se procesa un mensaje específico. Éste es uno de los métodos más fiables para validar el comportamiento de los filtros antes de su aplicación.

Trace proporciona visibilidad de:

- Cómo se analiza el mensaje
- Qué filtros se evalúan

- Si la condición se desencadena
- El orden de ejecución de la regla

Dado que ESA realiza el análisis MIME, la normalización de encabezados y la decodificación de contenido, el comportamiento dentro del dispositivo puede diferir de las herramientas de prueba de expresiones regulares externas. Para obtener instrucciones detalladas, los administradores deben consultar la documentación oficial de Cisco:

https://www.cisco.com/c/en/us/td/docs/security/ces/ces_16-0-3/user_guide/b_ESA_Admin_Guide_ces_16-0-3/b_ESA_Admin_Guide_12_1_chapter_0101001.html

El uso de Trace garantiza que el filtro se comporta como se espera dentro del motor de procesamiento real.

2. Creación del Filtro con una Acción de Registro

Otro enfoque seguro y recomendado es implementar el filtro con una acción no disruptiva, como el registro, en lugar de aplicar una acción agresiva como dejar caer, rebotar o poner en cuarentena los mensajes.

Al configurar el filtro para que registre una entrada cuando haya coincidencias, los administradores pueden:

- Observar la frecuencia de coincidencia
- Detectar desencadenadores inesperados
- Validar el impacto en el rendimiento
- Analizar el comportamiento del tráfico real

Este enfoque coloca eficazmente el filtro en una fase de supervisión controlada dentro del tráfico de producción. Una vez que se haya completado la validación suficiente y se haya confirmado que el comportamiento es correcto, la acción se puede cambiar de forma segura al modo de aplicación.

Introducción de la expresión en un filtro de contenido y en un diccionario

Una vez que la expresión regular se ha diseñado y validado correctamente, el siguiente paso es comprender cómo debe ingresarse dentro de Cisco ESA. La sintaxis puede ser ligeramente diferente dependiendo de si la expresión está configurada directamente en una condición de filtro de contenido o dentro de un diccionario. Esta diferencia a menudo causa confusión.

Introducción de la expresión en un filtro de contenido

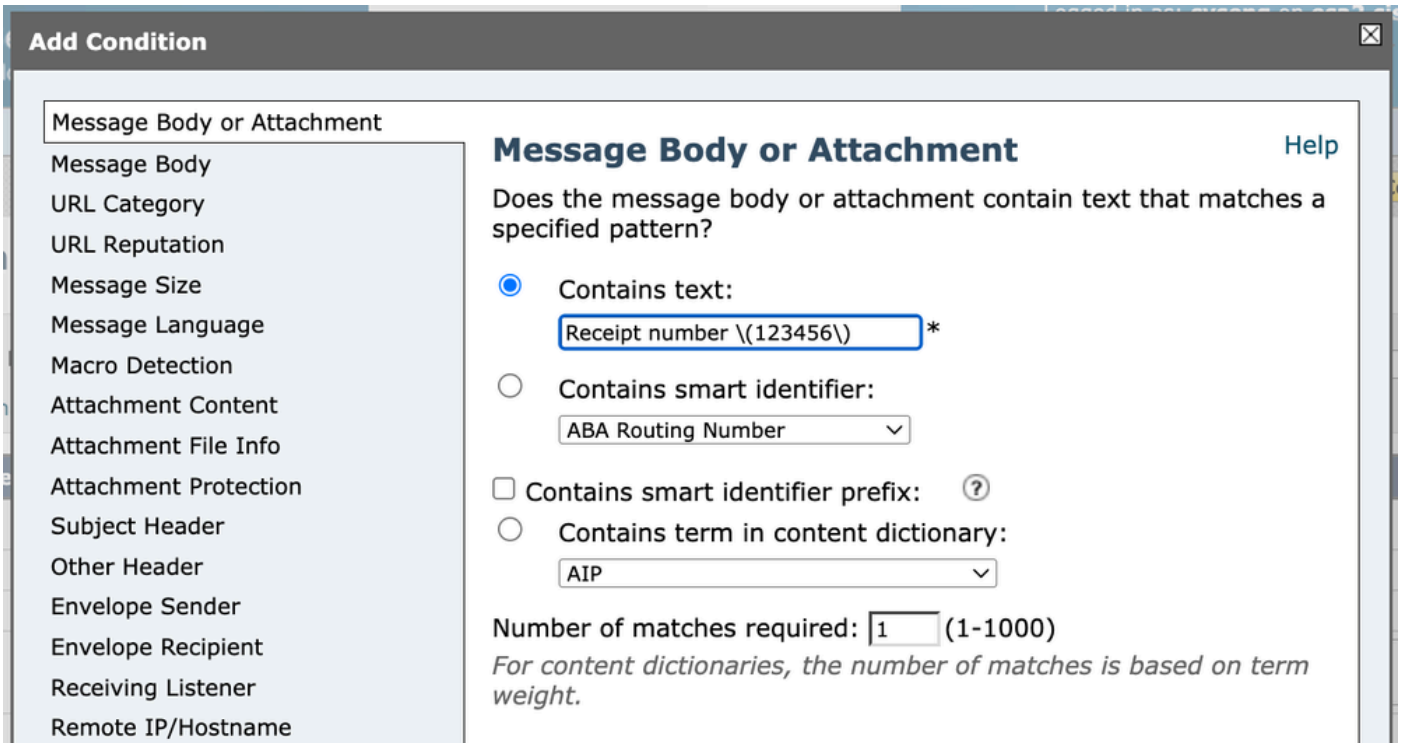
Al configurar una condición de filtro de contenido (por ejemplo, si coincide con el encabezado Asunto), se debe introducir la expresión regular en el campo de condición. Si queremos hacer coincidir el texto literal:

```
Receipt number (123456)
```

Debemos escapar de los paréntesis porque son caracteres especiales en expresiones regulares.

Por lo tanto, el regex en sí debe escribirse como:

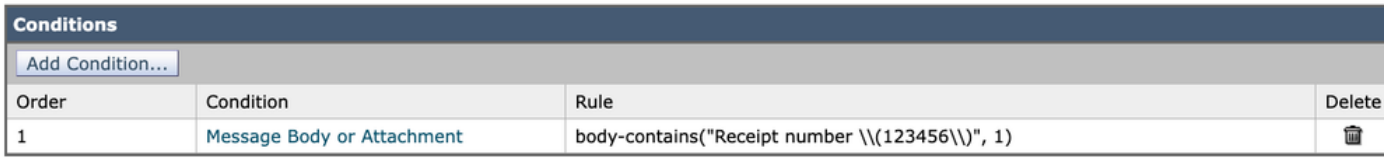
```
Receipt number \(123456\)
```



Filtro de contenido 1

Sin embargo, cuando se visualiza la condición de filtro completa en la GUI o en el resultado de la configuración avanzada, puede aparecer como:

```
subject == "Receipt number \\(123456\\)"
```



Filtro de contenido 2

Esto puede ser confuso a primera vista. La razón de las barras invertidas dobles (\\) es que la propia barra invertida es también un carácter especial dentro de las cadenas entrecomilladas. En este contexto, una barra invertida se utiliza para escapar del paréntesis para el motor regex, y la segunda barra invertida se utiliza para escapar de la barra invertida dentro de la cadena entrecomillada.

En la práctica:

\(123456\) es la expresión regular real.

\\(es como el sistema representa \(dentro de una cadena de configuración entrecomillada.

Aunque parece diferente cuando se muestra, la expresión regular lógica que se evalúa permanece:

Número de recibo \(123456\)

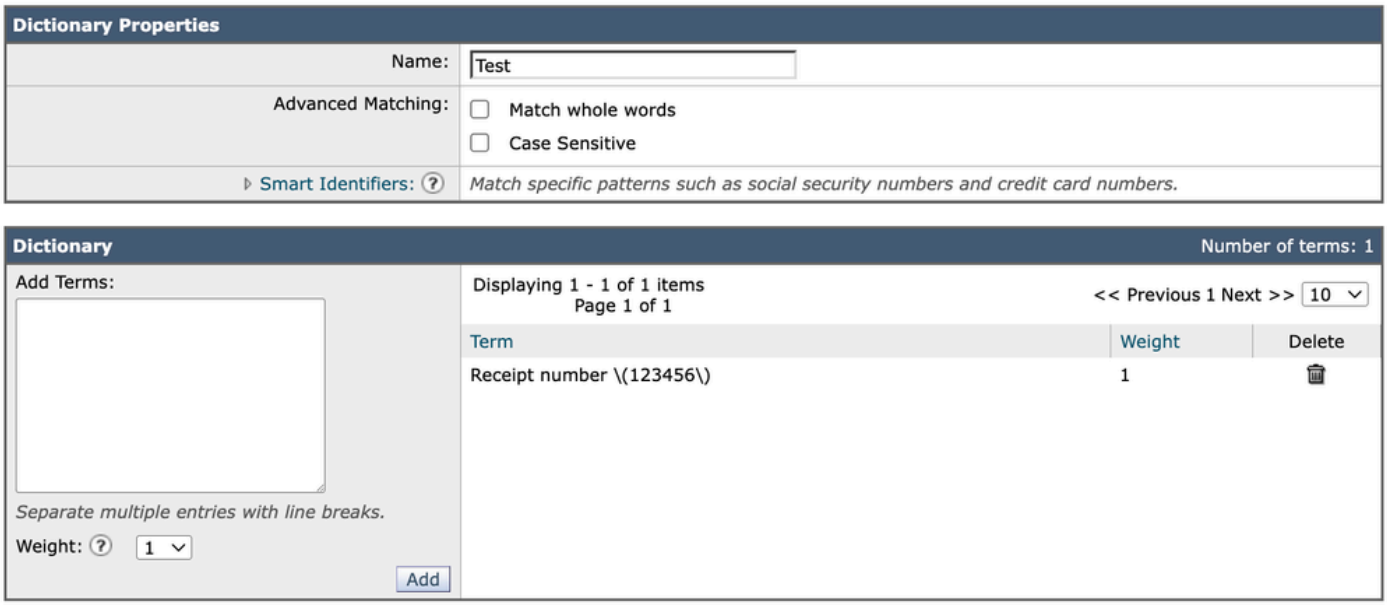
Esto es simplemente una cuestión de cadena escapando en el resultado de la configuración.

Introducción de la expresión en un diccionario

Al agregar la misma expresión a un diccionario, la entrada se introduce directamente como:

```
Receipt number \(123456\)
```

En este caso, se seguirá mostrando exactamente como está escrito. A diferencia de la representación GUI del filtro de contenido, los diccionarios no requieren capas de escape adicionales en su formato de configuración visual.



Diccionario

Cada entrada de diccionario se evalúa como texto sin formato o como una expresión regular dependiendo de su estructura. Si se incluyen caracteres especiales (como paréntesis en este caso), la expresión ya debe contener caracteres de escape cuando se introduzca.

Acerca de "Coincidir palabras enteras"

Al configurar un diccionario, hay una opción llamada "Coincidir palabras completas". En muchos casos, se recomienda no confiar en esta configuración cuando se trabaja con expresiones regulares.

La razón es que el comportamiento de límites de palabras se puede controlar con mayor precisión mediante anclajes regex.

Por ejemplo:

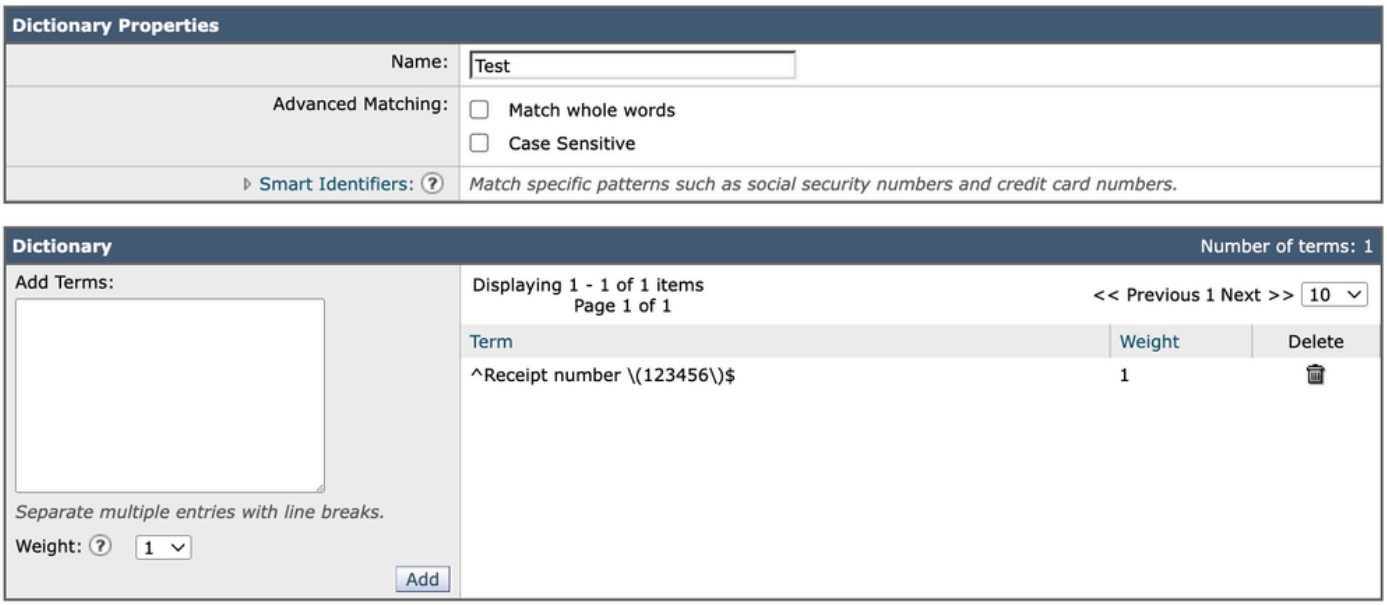
^ garantiza que la coincidencia comienza al principio.

$ garantiza que el partido termine al final.

Utilizar anclajes como:

```
^Receipt number \(123456\)$
```

Proporciona un control explícito y predecible sobre el comportamiento de coincidencia exacto. Este enfoque evita posibles ambigüedades relacionadas con la interpretación de los límites de las palabras, especialmente en entornos multilingües o no ASCII.



Diccionario 2

Por esta razón, es preferible administrar la precisión de coincidencia directamente dentro de la expresión regular en lugar de confiar en la opción "Coincidir palabras completas".

La comprensión de estas sutiles diferencias entre los filtros de contenido y los diccionarios garantiza que las expresiones se comporten de forma coherente y reduce el riesgo de errores de configuración durante la implementación.

Clasificación de costes Regex en Cisco ESA

Cuando se trabaja con expresiones regulares en Cisco ESA, el impacto en el rendimiento depende en gran medida de la cantidad de texto que debe escanear el motor y de la cantidad de retroceso que debe realizar. Dado que el ESA debe evaluar cuerpos de mensaje completos, partes MIME e incluso adjuntos descodificados, los patrones ineficientes pueden aumentar significativamente el uso de la CPU.

Se trata de una clasificación práctica que va del coste informático más alto al más bajo.

Patrones de alto riesgo más caros

Estas expresiones pueden afectar drásticamente al rendimiento, especialmente en mensajes de gran tamaño.

Cuantificadores anidados (peor caso)

Examples:

```
(.*)+
(.+)+
(\S+)+
```

Estos son extremadamente peligrosos porque crean escenarios de retroceso exponenciales.

Un cuantificador dentro de otro cuantificador fuerza al motor regex a probar muchas combinaciones antes de fallar.

En el tráfico real, esto puede causar graves picos de CPU.

Recomendación: Evite cuantificadores anidados sin límites y ambiguos.

Greedy .* Seguido de un patrón obligatorio

Ejemplo:

```
.*text
.*\/\?text
```

Este patrón primero consume todo el mensaje y después retrocede carácter por carácter hasta que encuentra la subcadena requerida.

Si el patrón no está presente —o aparece cerca del final— el motor retrocede y prueba el token requerido en muchas posiciones, lo que aumenta el costo de la CPU.

En ESA, donde los cuerpos pueden ser grandes e incluir contenido MIME, esto se vuelve costoso muy rápidamente.

Recomendación: No anteponga .* para detectar subcadenas. El ESA ya busca el contenido evaluado, y los comodines principales solo aumentan el seguimiento y el uso de la CPU.

```
text$
\/\?text$
```

Alternaciones grandes con prefijos compartidos

Ejemplo:

```
(a.*b|a.*c|a.*d)
```

Cuando varias alternativas comparten estructura, el motor evalúa cada rama secuencialmente.

Si las ramas tempranas casi coinciden pero fallan tarde, el motor se reintenta ampliamente.

Esto aumenta significativamente el tiempo de evaluación.

Coste medio: utilice con precaución

Estos patrones no son catastróficos pero pueden ser ineficientes.

Amplio .* Uso

Ejemplo:

```
https://.*\?text
```

Si bien no es exponencial, .* aún permite la coincidencia ilimitada. Si la subcadena esperada no aparece rápidamente, el motor explora grandes porciones del mensaje.

En ESA, esto es común cuando se analizan los cuerpos de correo electrónico en busca de URL de phishing.

Cuantificadores perezosos (+?, *?)

Ejemplo:

```
\S+?
.*?
```

Los cuantificadores perezosos cambian la estrategia de correspondencias (más corto primero). Pueden reducir la coincidencia excesiva en algunos patrones, pero en grandes cargas de trabajo de ‘búsqueda’ pueden aumentar los intentos cuando el token de terminación es tardío o falta.

En muchos casos prácticos de ESA, no proporcionan beneficios reales y pueden introducir reintentos internos innecesarios.

Clases de caracteres muy genéricos

Examples:

```
\S+
.+
```

Estos permiten un amplio rango de coincidencias, lo que aumenta el número de posibles rutas de

retroceso.

Siempre son preferibles clases de caracteres más específicas.

Bajo coste: patrones seguros y eficientes

Se recomiendan para entornos ESA de producción.

Literales fijos

Examples:

```
text
iw\.adc
```

Las cadenas literales son las coincidencias más eficientes posibles. El motor realiza comparaciones sencillas con una sobrecarga mínima.

Utilizar delimitadores para limitar el ámbito de búsqueda

Cuando la coincidencia se espera en una posición específica, considere anclar el patrón usando ^ o $. Los anclajes restringen la evaluación a posiciones fijas e impiden que el motor escanee todo el contenido innecesariamente. Esto puede reducir el retroceso y mejorar el rendimiento, especialmente en cuerpos de mensajes grandes o encabezados estructurados.

```
^Invoice$
```

Clases de caracteres específicas

```
[A-Za-z0-9.-]+
[^/\s]+
```

Estos restringen lo que puede coincidir, lo que reduce drásticamente el espacio de búsqueda y limita el retroceso.

Patrones estructurados y restringidos

Ejemplo:

```
https?:\/\/[A-Za-z0-9.-]+(?:\/[^?\s]*)*\/\?text
```

- El dominio es fijo.
- Sin uso de .*.
- no contiene patrones anidados catastróficos (ejemplo, (.*)+)
- Sin operadores perezosos innecesarios.
- Cada sección está restringida.

Esto reduce significativamente el impacto en la CPU en comparación con la coincidencia amplia de comodines.

Guía práctica para Cisco ESA

Al diseñar expresiones regulares para filtros de mensajes o de contenido:

1. Cuanto más específico sea el patrón, mejor será el rendimiento.
2. Evite .* a menos que sea realmente necesario — y especialmente evite colocar los tokens necesarios después de él.
3. No utilice nunca cuantificadores anidados.
4. Prefiera clases de caracteres explícitos en lugar de comodines.
5. Pruebe siempre las nuevas expresiones en el modo de supervisión antes de aplicarlas.

Comparación de rendimiento de Regex (contexto de Cisco ESA)

Patrón	Recomendado	Riesgo de retroceso	Impacto de ESA	Alternativa recomendada
https?:\/\/.*/\?text.*	No	Alto	Mayor	^https?:\/\/[A-Za-z0-9.-]+(?:\/[^?\s]*)*\/\?texto
https?:\/\/.*\?text	⚠ Con precaución	Medio-alto	Medio-alto	^https?:\/\/[^?\s]+\?text$
https?:\/\/.*	No	Medio-alto	Medio	^https?:\/\/[A-Za-z0-9.-]+(?:\/[^\s]*)*
.*password	No	Alto	Mayor	password$
.*texto.*	No	Alto	Mayor	texto
.*(factura|pago|transferencia)	No	Alto	Mayor	(factura|pago|transferencia)$

(.+)+	Nunca	Muy alta (exponencial)	Severo	Reestructurar sin cuantificadores anidados (ejemplo .+)
.*@.*	No	Alto	Mayor	[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}
\S+?	No es ideal	Medio	Medio	\S+ o una clase más específica como [A-Za-z0-9.-]+
.*\/admin	No	Alto	Mayor	\/admin$
.*(inicio de sesión|verificar).*	No	Alto	Mayor	(inicio de sesión|verificar)
^.*texto	No	Alto	Mayor	text$ (o ^text si la posición importa)

Conclusión

Las expresiones regulares son una herramienta potente y flexible dentro de Cisco ESA, que permite una inspección de contenido precisa y una aplicación de políticas avanzadas tanto en filtros de mensajes como en filtros de contenido. Sin embargo, con esa flexibilidad viene la responsabilidad. Las expresiones mal diseñadas o probadas de forma insuficiente pueden dar lugar a falsos positivos, detecciones perdidas, degradación del rendimiento o interrupción no intencionada del tráfico de correo electrónico legítimo.

Por esta razón, el uso de expresiones regulares en la ESA siempre debe ser un enfoque estructurado y disciplinado. La fase de creación debe garantizar que la expresión es sintácticamente correcta, escapada correctamente, eficiente y lógicamente alineada con el objetivo deseado. Las herramientas externas y la generación asistida por IA pueden acelerar significativamente este proceso, pero nunca deben reemplazar una validación cuidadosa.

Igualmente importante es la fase de validación dentro del propio entorno ESA. Debido a que ESA procesa los mensajes a través del análisis MIME, la normalización de encabezados y la decodificación de contenido, el comportamiento real puede diferir de las expectativas teóricas. El uso de herramientas como Trace e implementación de filtros inicialmente en el modo de registro o supervisión permite a los administradores confirmar el comportamiento correcto sin riesgo operativo.

En resumen, las expresiones regulares deben mantenerse lo más simples posible, probarse a fondo e implementarse con cautela. Un filtro bien diseñado y validado correctamente no solo

aplica la política de forma eficaz, sino que también protege la estabilidad del sistema y garantiza un comportamiento predecible en los entornos de producción.

Documentación

Para obtener información técnica adicional y orientación oficial sobre cómo se implementan y utilizan las expresiones regulares en Cisco ESA, los administradores deben consultar la documentación del producto de Cisco

La sección "Expresiones regulares en reglas" proporciona una descripción general de cómo se evalúan las expresiones regulares en los filtros de mensajes y los filtros de contenido, incluidas las consideraciones de sintaxis y el uso en condiciones de regla.

https://www.cisco.com/c/en/us/td/docs/security/ces/ces_16-0-3/user_guide/b_ESA_Admin_Guide_ces_16-0-3/b_ESA_Admin_Guide_12_1_chapter_01000.html#con_1192755

La sección "Directrices para el uso de expresiones regulares" ofrece recomendaciones prácticas sobre la sintaxis correcta, anclar expresiones, controlar caracteres especiales y evitar errores comunes que pueden afectar al rendimiento o a la precisión de coincidencia.

https://www.cisco.com/c/en/us/td/docs/security/ces/ces_16-0-3/user_guide/b_ESA_Admin_Guide_ces_16-0-3/b_ESA_Admin_Guide_12_1_chapter_01000.html#con_1125696

Se recomienda encarecidamente revisar estos recursos oficiales al diseñar o solucionar problemas de filtros que se basan en expresiones regulares, ya que proporcionan una orientación autorizada alineada con la versión específica de AsyncOS en uso.

Acerca de esta traducción
Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).