Configurar e validar expressões regulares no Cisco ESA e CES

Contents

Introdução

Este documento descreve como o ESA e o CES usam expressões regulares em filtros, as principais diferenças de comportamento e a necessidade de testes antes da aplicação.

Informações de Apoio

Este documento descreve como o Cisco Email Security Appliance (ESA) e o Cisco Cloud Email Security (CES) lidam com expressões regulares quando usados em filtros de mensagens e filtros de conteúdo. Ele se concentra especificamente em entender como as expressões regulares se comportam nesses componentes e como elas interagem com cabeçalhos de e-mail, conteúdo corporal e anexos.

É importante esclarecer desde o início que o mecanismo de expressão regular usado no módulo DLP se comporta de forma diferente. Portanto, tudo o que está descrito neste documento se aplica exclusivamente a filtros de mensagens e filtros de conteúdo e não se aplica a políticas DLP.

Ao trabalhar com expressões regulares no ESA, os administradores devem entender que o conteúdo do e-mail não é avaliado da mesma forma que é exibido visualmente em um cliente de e-mail. As mensagens de e-mail contêm informações de envelope, cabeçalhos estruturados, partes MIME e conteúdo potencialmente codificado. Como resultado, comparações realizadas por filtros podem produzir resultados inesperados se a estrutura da mensagem e o comportamento de regex não forem totalmente compreendidos.

Por esse motivo, qualquer novo filtro que use expressões regulares sempre pode ser habilitado no modo de monitor antes da aplicação. Isso permite a validação em relação ao tráfego real e evita o bloqueio não intencional ou o impacto no desempenho.

Dicionários e termos de pesquisa

Ao criar um filtro de mensagens ou um filtro de conteúdo, o termo inserido em muitas condições é interpretado como uma expressão regular. Este é um conceito crítico: mesmo quando o administrador pretende corresponder texto literal, o ESA pode processar a entrada usando a lógica regex.

Isso não se aplica uniformemente a todos os tipos de condição. Por exemplo, ao procurar um endereço IP específico em determinadas condições estruturadas, o valor não é interpretado como uma expressão regular. No entanto, ao pesquisar no cabeçalho Assunto, no corpo da mensagem, em um campo de cabeçalho específico ou em um nome de arquivo de anexo, o valor é normalmente tratado como um padrão regex.

Um exemplo comum ilustra isso claramente. Suponha que o objetivo seja bloquear e-mails com o assunto:

```
Receipt number (123456)
```

Como os parênteses são caracteres especiais em expressões regulares (usados para agrupamento), eles devem ter escape.

A expressão correta seria:

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

Se os parênteses não tiverem escape, o mecanismo regex os interpretará como operadores de agrupamento em vez de caracteres literais. Dependendo do padrão, isso pode causar correspondências não intencionais ou um comportamento diferente do esperado.

Por causa disso, é essencial entender quais caracteres têm significado especial em regex e garantir que eles sejam devidamente escapados quando a correspondência literal é necessária.

Exemplos de caracteres especiais e sua sintaxe de escape

A primeira coluna mostra um texto de exemplo contendo caracteres especiais, e a segunda coluna mostra como a sintaxe correta da expressão regular deve ser escrita para corresponder ao texto literal no Cisco ESA (regex estilo Python).

Texto literal a corresponder	Sintaxe de expressão regular correta
Número do recibo (123456)	Número do recibo \(123456\)
user@example.com	user@example.com
www.test.abc	www\.test\.abc
file_name.txt	file_name\.txt
o preço é 10,50	o preço é 10\,50
C:\Users\Admin	C:\\Users\\Admin
[CONFIDENCIAL]	\[CONFIDENCIAL\]
{fatura}	\{fatura\}
+34 600 123 456	\+34 600 123 456
pergunta?	pergunta\?
100% garantido	100% garantido (% não exige saída)
símbolo asterisco *	símbolo de asterisco *
A|B	A|B
circunflexo ^start	acento circunflexo \^start
dólar US$ 100	dólar \$100

Limitação do Uso de Expressões Regulares

As expressões regulares devem ser usadas com cuidado e somente quando necessário. Embora forneçam recursos de correspondência poderosos, expressões excessivas ou mal projetadas podem aumentar o tempo de processamento de mensagens e produzir correspondências não intencionais.

Uma construção particular que requer cuidado é .* , que representa "qualquer caractere, zero ou mais vezes". Quando colocado no início ou no final de uma expressão, pode causar backtracking excessivo e sobrecarga de processamento desnecessária.

A documentação da Cisco indica que as entradas que usam .* no início ou no final podem fazer com que o sistema seja bloqueado sob certas condições ao corresponder a partes MIME específicas. Por esse motivo, a Cisco recomenda evitar o uso de .* à esquerda ou à direita sempre que possível.

Em muitos cenários, os administradores usam padrões como .*fatura.*quando poderiam simplesmente gravar a fatura e produzir o mesmo resultado prático no ESA. Como o mecanismo de varredura já pesquisa as áreas de conteúdo relevantes, cercar uma palavra com .* é normalmente redundante e ineficiente em termos computacionais.

Caution: A recomendação geral é manter expressões regulares tão simples e precisas quanto possível.

Filtros de mensagens, filtros de conteúdo e dicionários

O Cisco ESA fornece vários mecanismos para avaliar mensagens e aplicar ações. Os filtros de mensagens operam no início do pipeline e usam uma sintaxe de estilo de script. Eles são extremamente flexíveis e permitem uma lógica avançada que envolve dados de envelope, cabeçalhos e propriedades de anexo. No entanto, como são executados no início da cadeia de processamento, os filtros de mensagem ineficientes podem afetar negativamente o desempenho.

Os filtros de conteúdo são configurados por meio da interface gráfica e operam depois que a mensagem é aceita. Para a maioria dos casos de uso de inspeção de conteúdo, os filtros de conteúdo são mais fáceis de gerenciar e mais seguros do ponto de vista do desempenho.

Tanto em filtros de mensagens quanto em filtros de conteúdo, expressões regulares podem ser introduzidas diretamente em uma condição ou indiretamente por meio do uso de dicionários.

Os dicionários permitem que os administradores centralizem termos de pesquisa reutilizáveis. Cada entrada é escrita em uma linha separada e pode ser um texto simples ou uma expressão regular. Os dicionários também suportam caracteres não-ASCII, tornando-os adequados para ambientes multilíngues.

Em algumas situações, certas construções complexas de expressões regulares não podem se comportar de forma idêntica dentro dos dicionários. Quando isso ocorre, a expressão regular deve ser colocada diretamente na condição de filtro em vez de dentro do dicionário.

O Cisco ESA permite a criação de até 150 dicionários de conteúdo. Por padrão, 100 dicionários podem ser configurados, a menos que o limite seja modificado via CLI usando o comando dictionaryconfig.

Os dicionários também podem implementar a ponderação de termos. Cada termo pode receber um peso numérico e, quando o ESA examina uma mensagem, ele multiplica o número de ocorrências desse termo pelo seu peso. A pontuação resultante é comparada a um limite definido no filtro. Esse modelo de pontuação permite uma aplicação de política mais flexível e gradual.

Além disso, os dicionários podem incluir Smart Identifiers, que são detectores algorítmicos de padrões numéricos estruturados, como números de previdência social ou identificadores bancários.

Mecanismo de expressão regular

O Cisco ESA usa expressões regulares baseadas no estilo de módulo Python re. Embora isso forneça compatibilidade com a sintaxe regex Python comum, nem todos os recursos avançados suportados em ambientes Python completos são necessariamente suportados no ESA.

Para correspondência de string exata, as expressões devem ser ancoradas usando ^ no início e $ no final. Sem essas âncoras, o mecanismo regex pode combinar substrings em vez de valores completos.

Por exemplo, a expressão:

```
sun.com
```

Corresponder strings como:

```
thegodsunocommando
```

No entanto, a expressão:

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

Corresponde somente à string exata sun.com.

Ao corresponder uma string vazia, é importante não usar "", já que isso corresponde efetivamente a todas as strings. Em vez disso, a expressão correta é:

```
^$
```

Como o Cisco ESA usa expressões regulares no estilo Python, há algumas maneiras de fazer uma comparação que não diferencia maiúsculas de minúsculas.

Por padrão, como mencionado, as expressões regulares diferenciam maiúsculas de minúsculas. Isso significa procurar:

```
foo
```

Só combine foo, mas não foo, foo ou foo.

Se você deseja executar uma correspondência que não diferencia maiúsculas de minúsculas, você pode usar o sinalizador embutido (?i) no início da expressão regular. Isto diz ao mecanismo regex para ignorar o caso para o resto do padrão.

Por exemplo:

```
(?i)foo
```

Esta expressão corresponde a:

- foo
- FOO
- Foo
- Of

Se você quiser corresponder a string inteira exatamente, ignorando maiúsculas e minúsculas, você pode combinar o sinalizador que não diferencia maiúsculas de minúsculas com âncoras:

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

Isso garante que o valor total seja exatamente "foo", independentemente da capitalização.

Outra alternativa (menos prática) seria definir explicitamente todas as combinações possíveis

usando classes de caracteres, por exemplo:

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

No entanto, essa abordagem torna-se difícil de manter e não é recomendada quando o sinalizador (?i) pode ser usado.

Na maioria dos cenários ESA, o método preferido e mais limpo para correspondência que não diferencia maiúsculas de minúsculas é usar:

```
(?i)
```

no início da expressão regular.

Caracteres não ASCII e limites de palavras

Em idiomas que usam conjuntos de caracteres de byte duplo, os conceitos de limites de palavras ou maiúsculas e minúsculas não podem se comportar como esperado. Expressões complexas que dependem de construções como \w podem produzir resultados inconsistentes quando a codificação ou localidade é desconhecida.

Nesses casos, pode ser aconselhável desativar a aplicação de limite de palavra na configuração do dicionário ou simplificar a expressão para evitar dependência de classes de caracteres ambíguas.

Ao trabalhar com dicionários não ASCII, a exibição CLI não pode renderizar caracteres corretamente, dependendo da codificação do terminal. Nesses casos, a abordagem recomendada é exportar o dicionário para um arquivo de texto, editá-lo externamente e importá-lo novamente.

Gravando Filtros Eficientes

A eficiência é essencial ao gravar filtros, especialmente em ambientes de alto volume. Um erro comum é escrever longas cadeias de condições OR para correspondências semelhantes.

Por exemplo, verificar dezenas de extensões de anexo individualmente força o mecanismo regex a inicializar repetidamente. Isso aumenta o uso da CPU e reduz a capacidade de manutenção.

Em vez de gravar muitas comparações separadas, agrupá-las usando alternação dentro de uma única expressão regular reduz significativamente a sobrecarga de processamento. Isso reduz o número de vezes que o mecanismo regex é chamado e facilita a manutenção do filtro.

O design eficiente do filtro não se trata apenas de legibilidade, ele afeta diretamente o

desempenho do sistema.

PDFs e expressões regulares

A correspondência de conteúdo dentro de arquivos PDF pode produzir resultados inesperados, dependendo de como o PDF foi gerado. Alguns PDFs não contêm espaços lógicos ou quebras de linha em sua representação interna. O mecanismo de varredura tenta reconstruir o espaçamento lógico com base no posicionamento da palavra.

Se uma palavra for construída usando várias fontes ou tamanhos de fonte, a representação interna poderá fragmentar o texto. Por exemplo, a palavra "callout" pode ser interpretada internamente como "chamar" ou "c a l lout".

Nesses casos, a tentativa de corresponder a expressão "callout" pode falhar porque a representação interna não contém aquela string contígua exata. Os administradores devem estar cientes dessa limitação ao projetar políticas baseadas em conteúdo direcionadas a anexos de PDF.

Testando Expressões Regulares

Testar expressões regulares antes de implantá-las na produção é um requisito operacional crítico. Uma expressão regular que aparece sintaticamente correta pode se comportar de forma muito diferente quando avaliada em relação ao tráfego de e-mail real. Sem testes adequados, um filtro pode gerar falsos positivos, falhar na detecção de padrões pretendidos, introduzir sobrecarga de desempenho ou interromper involuntariamente o fluxo de e-mail legítimo.

Os testes devem ser abordados como um processo estruturado, em duas etapas, para minimizar o risco antes de permitir um filtro na produção.

Fase 1 - Projeto e validação de expressões regulares
A primeira fase se concentra em projetar e validar a própria expressão regular antes de integrá-la ao Cisco ESA.

1. Uso de regex101 ou ferramentas similares

Plataformas online, como http://regex101.com (ou ferramentas equivalentes) são altamente úteis durante a fase de projeto. Ao usar essas ferramentas, o tipo Python deve ser selecionado para aproximar o mecanismo regex da ESA.

Essas plataformas permitem que os administradores:

- Validar correção da sintaxe
- Confirme se os caracteres especiais têm escape adequado
- Testar casos correspondentes e não correspondentes

- Visualizar comportamento de agrupamento e quantificador
- Identificar construções potencialmente gananciosas como .*

No entanto, essas ferramentas simulam o comportamento padrão de regex Python e podem suportar recursos não totalmente implementados no Cisco ESA. Por conseguinte, devem ser considerados instrumentos de validação preliminares e não testes de compatibilidade definitivos.

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

Assistentes baseados em IA podem acelerar a criação de regex, especialmente para cenários de correspondência complexos. Ao descrever o comportamento desejado em linguagem natural, os administradores podem obter uma proposta inicial de regex que pode ser refinada.

As ferramentas de IA são particularmente úteis para:

- Gerando expressões agrupadas complexas
- Convertendo requisitos de negócios em sintaxe regex
- Simplificação de condições longas baseadas em OR em alternações agrupadas

No entanto, as expressões geradas pela IA devem ser sempre revistas de forma crítica. Eles podem apresentar ineficiências, construções sem suporte ou lógica excessivamente complexa. A ajuda à IA deve ser tratada como um auxílio à redação e não como uma validação final. Cada expressão gerada por IA ainda deve ser testada usando métodos de validação estruturados.

Fase 2 - Filtrar a validação do comportamento no Cisco ESA

Depois que a expressão tiver sido validada, a segunda fase se concentra em confirmar como ela se comporta dentro do Cisco ESA quando aplicada ao processamento real de mensagens.

1. Usando o Recurso de Rastreamento no Console CES

O recurso Rastrear no console do Cisco Email Security (CES) permite que os administradores simulem e analisem como uma mensagem específica é processada. Este é um dos métodos mais confiáveis para validar o comportamento do filtro antes da aplicação.

O Trace oferece visibilidade em:

- Como a mensagem é analisada
- Quais filtros são avaliados
- Se a condição é disparada
- A ordem de execução da regra

Como o ESA executa análise MIME, normalização de cabeçalho e decodificação de conteúdo, o comportamento dentro do dispositivo pode ser diferente das ferramentas de teste regex externas.

Para obter instruções detalhadas, os administradores devem consultar a documentação oficial da 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

O uso de Rastreamento garante que o filtro se comporte como esperado dentro do mecanismo de processamento real.

2. Criando o Filtro com uma Ação de Log

Outra abordagem segura e recomendada é implantar o filtro com uma ação sem interrupções, como registro, em vez de aplicar uma ação agressiva como descartar, devolver ou colocar mensagens em quarentena.

Ao configurar o filtro para registrar uma entrada quando houver correspondência, os administradores podem:

- Observar a frequência de correspondência
- Detectar disparadores inesperados
- Validar impacto no desempenho
- Analisar o comportamento real do tráfego

Essa abordagem coloca efetivamente o filtro em uma fase de monitoramento controlado dentro do tráfego de produção. Quando uma validação suficiente for concluída e o comportamento for confirmado como correto, a ação poderá ser alterada com segurança para o modo de imposição.

Introdução à Expressão em um Filtro de Conteúdo e em um Dicionário

Depois que a expressão regular tiver sido projetada e validada corretamente, a próxima etapa é entender como ela deve ser inserida no Cisco ESA. A sintaxe pode ser ligeiramente diferente, dependendo se a expressão estiver configurada diretamente em uma condição de Filtro de conteúdo ou dentro de um Dicionário. Essa diferença frequentemente causa confusão.

Introdução à Expressão em um Filtro de Conteúdo

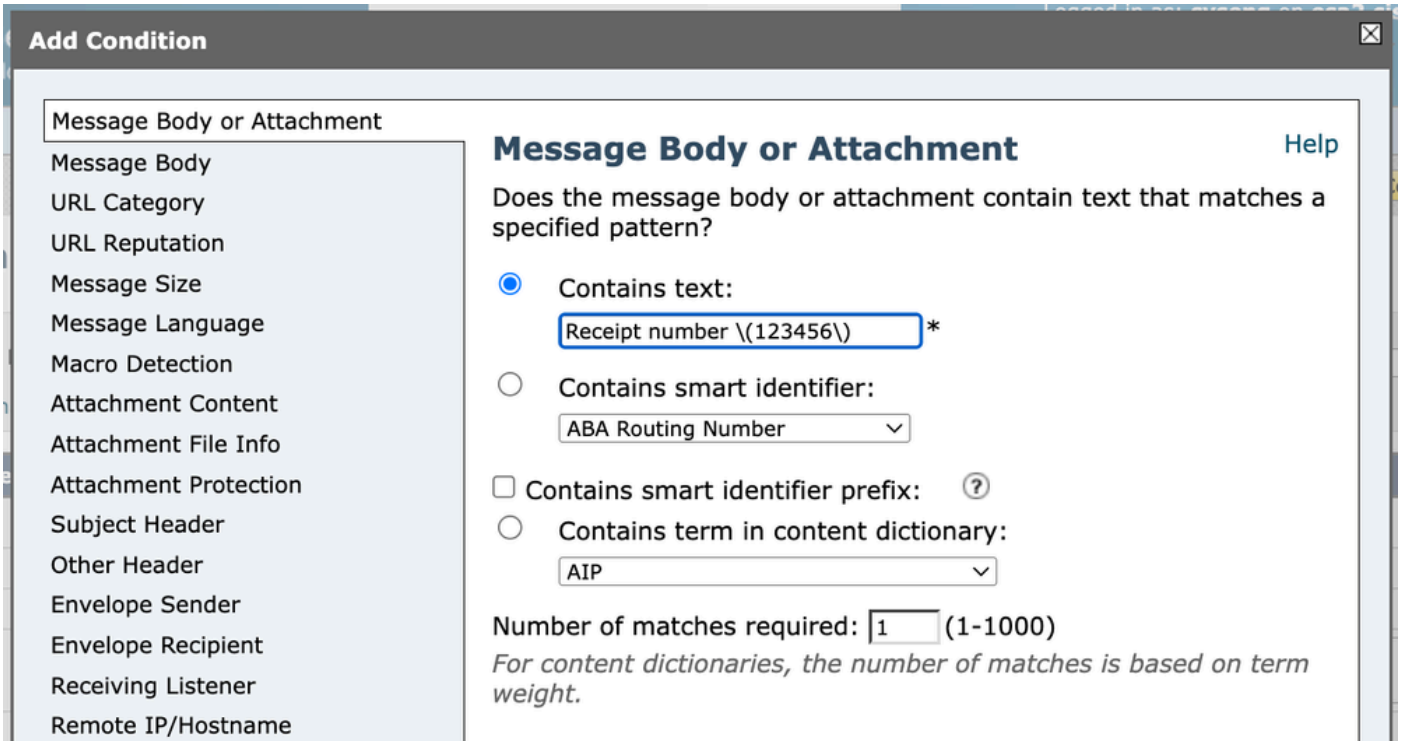
Ao configurar uma condição de Filtro de conteúdo (por exemplo, correspondência com o cabeçalho Assunto), a expressão regular deve ser informada no campo de condição. Se quisermos corresponder o texto literal:

```
Receipt number (123456)
```

Precisamos escapar os parênteses porque eles são caracteres especiais em expressões regulares.

Portanto, o próprio regex deve ser escrito como:

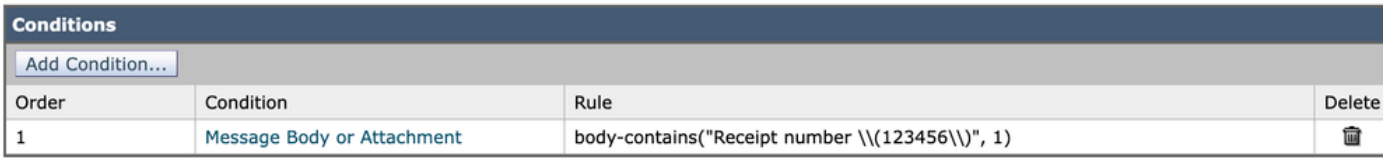
```
Receipt number \(123456\)
```



Filtro de conteúdo 1

No entanto, ao visualizar a condição de filtro completo na saída da GUI ou da configuração avançada, ela pode aparecer como:

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



Filtro de conteúdo 2

À primeira vista, isso pode ser confuso. A razão para as barras invertidas duplas (\\) é que a própria barra invertida também é um caractere especial dentro de strings entre aspas. Neste

contexto, uma barra invertida é usada para escapar do parêntese para o mecanismo regex, e a segunda barra invertida é usada para escapar a barra invertida dentro da string entre aspas.

Em termos práticos:

\(123456\) é a expressão regular real.

\\(é como o sistema representa \(dentro de uma string de configuração entre aspas.

Embora pareça diferente quando exibida, a expressão regular lógica que está sendo avaliada permanece:

Número do recibo \(123456\)

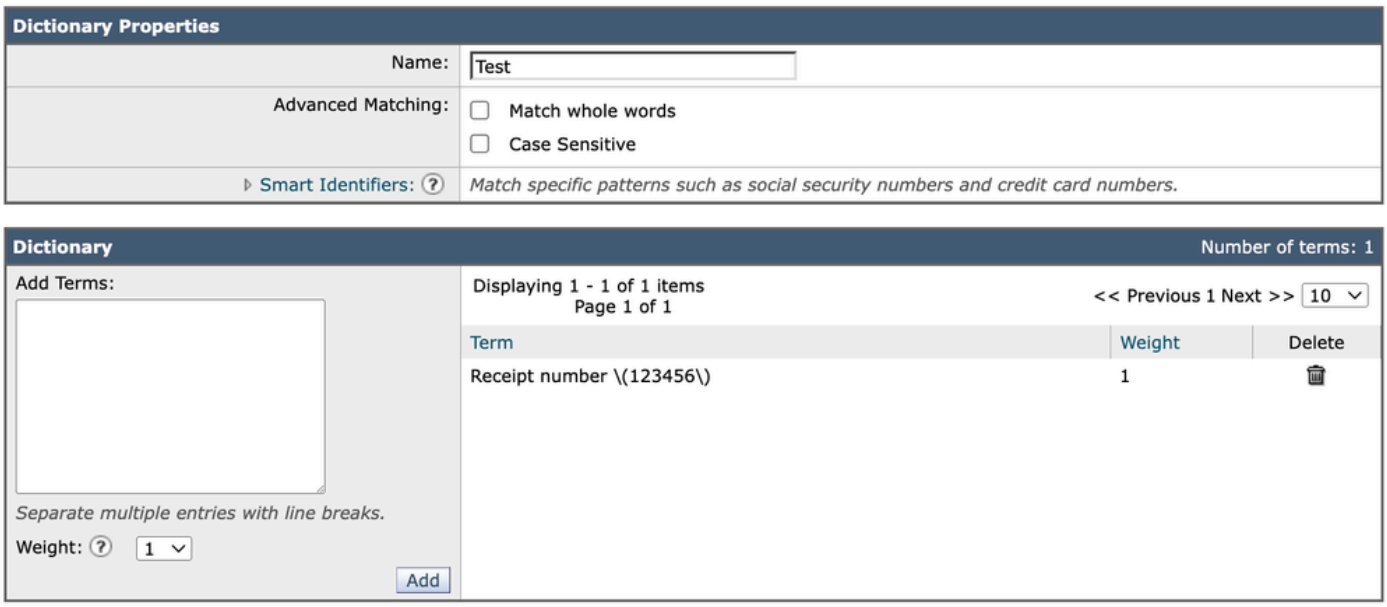
Isso é simplesmente uma questão de escape de string na saída de configuração.

Apresentando a Expressão em um Dicionário

Ao adicionar a mesma expressão a um dicionário, a entrada é introduzida diretamente como:

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

Nesse caso, ele continua a ser exibido exatamente como está escrito. Diferentemente da representação da GUI do filtro de conteúdo, os dicionários não exigem camadas de saída adicionais em seu formato de configuração visual.



Dicionário

Cada entrada de dicionário é avaliada como texto sem formatação ou como uma expressão regular, dependendo de sua estrutura. Se caracteres especiais forem incluídos (como parênteses neste caso), a expressão já deverá ter escapado corretamente quando inserida.

Sobre "Match Whole Words"

Ao configurar um dicionário, há uma opção chamada "Coincidir palavras inteiras". Em muitos casos, é recomendável não confiar nessa configuração ao trabalhar com expressões regulares.

A razão é que o comportamento de limite de palavra pode ser controlado mais precisamente usando âncoras regex.

Por exemplo:

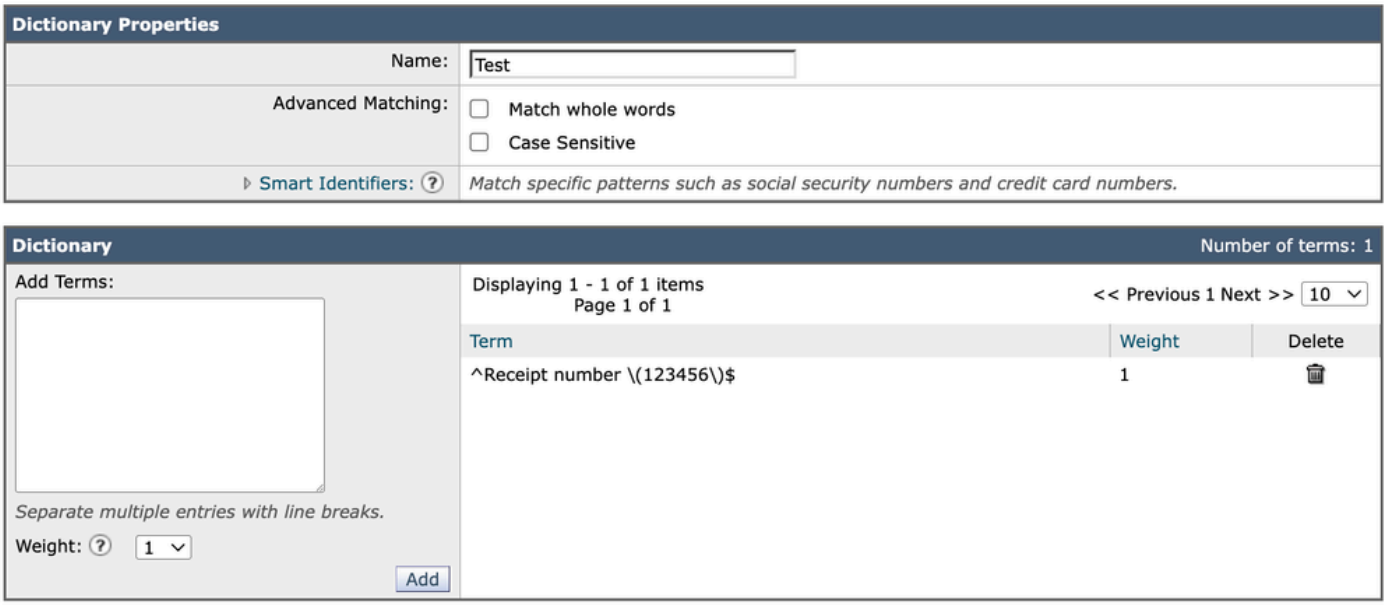
^ garante que a partida comece no início.

$ garante que a correspondência termine no final.

Usando âncoras como:

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

Fornece controle explícito e previsível sobre o comportamento de correspondência exata. Essa abordagem evita possíveis ambiguidades relacionadas à forma como os limites de palavras são interpretados, especialmente em ambientes multilíngues ou não ASCII.



Dicionário 2

Por esse motivo, geralmente é preferível gerenciar a precisão de correspondência diretamente na expressão regular em vez de depender da opção "Corresponder palavras inteiras".

Compreender essas diferenças sutis entre filtros de conteúdo e dicionários garante que as expressões se comportem de forma consistente e reduz o risco de erros de configuração durante a implementação.

Classificação de custo Regex no Cisco ESA

Ao trabalhar com expressões regulares no Cisco ESA, o impacto no desempenho depende em grande parte de quanto texto o mecanismo deve digitalizar e quanto backtracking ele deve executar. Como o ESA deve avaliar corpos inteiros de mensagens, partes MIME e até mesmo anexos decodificados, padrões ineficientes podem aumentar significativamente o uso da CPU.

É uma classificação prática do mais alto custo computacional para o mais baixo.

Mais caro — Padrões de alto risco

Essas expressões podem impactar consideravelmente o desempenho, especialmente em mensagens grandes.

Quantificadores Aninhados (Pior Caso)

Examples:

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

Eles são extremamente perigosos, pois criam cenários de retrocesso exponenciais.

Um quantificador dentro de outro quantificador força o mecanismo regex a tentar muitas combinações antes de falhar.

No tráfego real, isso pode causar sérios picos de CPU.

Recomendação: Evite quantificadores aninhados ilimitados e ambíguos.

Ganancioso .* Seguido por um Padrão Necessário

Exemplo:

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

Esse padrão primeiro consome a mensagem inteira e, em seguida, volta a rastrear caractere por caractere até encontrar a substring necessária.

Se o padrão não estiver presente — ou aparecer próximo ao final — o mecanismo rastreia novamente e testa o token necessário em muitas posições, o que aumenta o custo da CPU.

Na ESA, onde os corpos podem ser grandes e incluir conteúdo MIME, isso se torna caro muito rapidamente.

Recomendação: Não anexe .* para detectar substrings. O ESA já pesquisa o conteúdo avaliado, e os curingas líderes aumentam apenas o backtracking e o uso da CPU.

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

Grandes Alterações com Prefixos Compartilhados

Exemplo:

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

Quando várias alternativas compartilham estrutura, o mecanismo avalia cada filial sequencialmente.

Se as ramificações iniciais quase coincidirem, mas falharem com atraso, o mecanismo tentará novamente extensivamente.

Isso aumenta significativamente o tempo de avaliação.

Custo Médio — Use Com Cuidado

Esses padrões não são catastróficos, mas ainda podem ser ineficientes.

Amplo .* Uso

Exemplo:

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

Embora não exponencial, .* ainda permite correspondência ilimitada. Se a substring esperada não aparecer rapidamente, o mecanismo verificará grandes partes da mensagem.

No ESA, isso é comum durante a varredura de URLs de phishing no corpo do e-mail.

Quantificadores ociosos (+?, *?)

Exemplo:

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

Quantificadores lentos alteram a estratégia de correspondência (shortest-first). Eles podem reduzir a sobrecorrespondência em alguns padrões, mas em grandes cargas de trabalho de ‘pesquisa’ podem aumentar as tentativas quando o token de terminação está atrasado ou ausente.

Em muitos casos de uso do ESA, eles não oferecem benefícios reais e podem introduzir novas tentativas internas desnecessárias.

Classes de Caracteres Muito Genéricos

Examples:

```
\S+
.+
```

Eles permitem uma ampla faixa de correspondência, aumentando o número de caminhos de backtracking em potencial.

Classes de caracteres mais específicas são sempre preferíveis.

Baixo custo — padrões seguros e eficientes

Eles são recomendados para ambientes ESA de produção.

Literais fixos

Examples:

```
text
iw\.adc
```

As strings literais são as correspondências mais eficientes possíveis. O mecanismo executa comparações diretas com sobrecarga mínima.

Usar âncoras para limitar o escopo da pesquisa

Quando a correspondência for esperada em uma posição específica, considere ancorar o padrão usando ^ ou $. As âncoras restringem a avaliação a posições fixas e impedem que o mecanismo examine todo o conteúdo desnecessariamente. Isso pode reduzir o backtracking e melhorar o desempenho, principalmente em grandes corpos de mensagens ou cabeçalhos estruturados.

```
^Invoice$
```

Classes de caracteres específicos

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

Eles restringem o que pode ser correspondido, reduzindo drasticamente o espaço de pesquisa e limitando o backtracking.

Padrões estruturados e restritos

Exemplo:

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

- O domínio é fixo.
- Sem uso de .*.
- não contém padrões aninhados catastróficos (exemplo, (.*)+)
- Sem operadores preguiçosos desnecessários.
- Cada seção é restrita.

Isso reduz significativamente o impacto da CPU em comparação com a correspondência de caractere geral.

Orientação prática para Cisco ESA

Ao projetar regex para filtros de mensagem ou conteúdo:

1. Quanto mais específico for o padrão, melhor será o desempenho.
2. Evite .* a menos que seja realmente necessário — e, especialmente, evite colocar os tokens necessários depois dele.
3. Nunca use quantificadores aninhados.
4. Prefira classes de caracteres explícitas a curingas.

5. Sempre teste novas expressões no modo de monitor antes da aplicação.

Comparação de desempenho Regex (contexto Cisco ESA)

Padrão	Recomendado	Risco de retrocesso	Impacto do ESA	Alternativa recomendada
https:\/\/.*/\?texto.*	No	Alto	Mais alto	^https?:\/\/[A-Za-z0-9.-]+(?:\/[^?\s]*)*\/\?texto
https:\/\/.*\?texto	⚠ Com cuidado	Médio-alto	Médio-alto	^https?:\/\/[^?\s]+\?texto$
https:\/\/.*	No	Médio-alto	Médio	^https?:\/\/[A-Za-z0-9.-]+(?:\/[^\s]*)*
.*senha	No	Alto	Mais alto	senha$
.*texto.*	No	Alto	Mais alto	texto
.*(fatura|pagamento|transferência)	No	Alto	Mais alto	(fatura|pagamento|transferência)
(.+)+	Nunca	Muito Alta (Exponencial)	Severe	Reestruturar sem quantificadores aninhados (exemplo .+)
.*@.*	No	Alto	Mais alto	[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}
\S+?	Não ideal	Médio	Médio	\S+ ou classe mais específica como [A-Za-z0-9.-]+
.*\/admin	No	Alto	Mais alto	\/admin$
.*(login|verificar).*	No	Alto	Mais	(login|verificar)

			alto	
^.*texto	No	Alto	Mais alto	text$ (ou ^text se a posição for importante)

Conclusão

As expressões regulares são uma ferramenta poderosa e flexível dentro do Cisco ESA, permitindo inspeção precisa de conteúdo e aplicação avançada de políticas em filtros de mensagens e filtros de conteúdo. No entanto, essa flexibilidade traz responsabilidades. Expressões mal projetadas ou insuficientemente testadas podem levar a falsos positivos, detecções perdidas, degradação de desempenho ou interrupção não intencional do tráfego de e-mail legítimo.

Por esta razão, a utilização de expressões regulares no SEC deve ser sempre uma abordagem estruturada e disciplinada. A fase de criação deve garantir que a expressão seja sintaticamente correta, adequadamente escapada, eficiente e logicamente alinhada com o objetivo pretendido. Ferramentas externas e geração assistida por IA podem acelerar significativamente esse processo, mas nunca devem substituir a validação cuidadosa.

Igualmente importante é a fase de validação no próprio ambiente do SEC. Como o ESA processa mensagens através de análise MIME, normalização de cabeçalho e decodificação de conteúdo, o comportamento real pode diferir das expectativas teóricas. Usar ferramentas como Rastrear e implantar filtros inicialmente no modo de registro ou monitoramento permite que os administradores confirmem o comportamento correto sem risco operacional.

Em resumo, as expressões regulares devem ser mantidas o mais simples possível, testadas minuciosamente e implantadas com cautela. Um filtro bem projetado e devidamente validado não apenas aplica a política de forma eficaz, mas também protege a estabilidade do sistema e garante um comportamento previsível nos ambientes de produção.

Documentação

Para obter detalhes técnicos adicionais e orientação oficial sobre como as expressões regulares são implementadas e usadas no Cisco ESA, os administradores devem consultar a documentação do produto Cisco

A seção "Expressões regulares em regras" fornece uma visão geral de como as expressões regulares são avaliadas em Filtros de mensagens e Filtros de conteúdo, incluindo considerações de sintaxe e uso em condições de regra.

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

A seção "Diretrizes para usar expressões regulares" oferece recomendações práticas sobre sintaxe correta, expressões de ancoragem, tratamento de caracteres especiais e evitar erros comuns que podem afetar o desempenho ou a precisão da correspondência.

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

É altamente recomendável revisar esses recursos oficiais ao projetar ou solucionar problemas de filtros que dependem de expressões regulares, pois eles fornecem orientação autoritativa alinhada com a versão específica do AsyncOS em uso.

Sobre esta tradução
A Cisco traduziu este documento com a ajuda de tecnologias de tradução automática e humana para oferecer conteúdo de suporte aos seus usuários no seu próprio idioma, independentemente da localização.

Observe que mesmo a melhor tradução automática não será tão precisa quanto as realizadas por um tradutor profissional.

A Cisco Systems, Inc. não se responsabiliza pela precisão destas traduções e recomenda que o documento original em inglês (link fornecido) seja sempre consultado.