Cisco ESAおよびCESでの正規表現の設定と検証

内容

はじめに

このドキュメントでは、ESAとCESがフィルタで正規表現を使用する方法、主な動作の違い、および適用前のテストの必要性について説明します。

バックグラウンド情報

このドキュメントでは、メッセージフィルタおよびコンテンツフィルタ内でCisco Eメールセキュリティアプライアンス(ESA)およびCiscoクラウドEメールセキュリティ(CES)を使用する際の正規表現の処理方法について説明します。特に、これらのコンポーネントでの正規表現の動作と、電子メールヘッダー、本文の内容、および添付ファイルとの相互作用について理解することに重点を置いています。

DLPモジュールで使用される正規表現エンジンの動作が最初から異なることを明確にすることが重要です。したがって、このドキュメントで説明している内容はすべて、メッセージフィルタとコンテンツフィルタにのみ適用され、DLPポリシーには適用されません。

ESAで正規表現を使用して作業する場合、管理者は、電子メールコンテンツがメールクライアントで視覚的に表示されるのと同じように評価されないことを理解する必要があります。電子メールメッセージには、エンベロープ情報、構造化ヘッダー、MIMEパーツ、およびエンコードされた可能性のあるコンテンツが含まれます。その結果、メッセージ構造と正規表現の動作が完全に理解されていない場合、フィルタによって実行される比較によって、予期しない結果が生じる可能性があります。

このため、正規表現を使用する新しいフィルタは、適用する前にモニタモードで常に有効にできます。これにより、実際のトラフィックに対する検証が可能になり、意図しないブロッキングやパフォーマンスへの影響を防ぐことができます。

辞書と検索語

メッセージフィルタまたはコンテンツフィルタを作成する場合、多くの条件で入力される用語は正規表現として解釈されます。これは重要な概念です。管理者がリテラルテキストを照合する場合でも、ESAは正規表現ロジックを使用して入力を処理できます。

これは、すべての条件タイプに一様に適用されるわけではありません。たとえば、特定の構造化条件で特定のIPアドレスを検索する場合、値は正規表現として解釈されません。ただし、件名ヘッダー、メッセージ本文、特定のヘッダーフィールド、または添付ファイル名で検索する場合、通常、値は正規表現パターンとして扱われます。

一般的な例でこれを明確に説明します。次の件名の電子メールをブロックすることを目標としていると仮定します。

```
Receipt number (123456)
```

カッコは（グループ化に使用される）正規表現の特殊文字であるため、エスケープする必要があります。

正しい式は次のとおりです。

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

カッコがエスケープされない場合、正規表現エンジンはリテラル文字ではなくグループ化演算子として解釈します。パターンによっては、予期しない一致や予期しない動作が発生する可能性があります。

このため、正規表現内で特殊な意味を持つ文字を理解し、リテラル照合が必要な場合に正しくエスケープされるようにすることが不可欠です。

特殊文字とエスケープされた構文の例

最初の列は特殊文字を含むテキストのサンプルを示し、2番目の列はCisco ESA（Pythonスタイルの正規表現）内のリテラルテキストと一致するように正しい正規表現の構文を記述する方法を示しています。

一致させるリテラルテキスト	正しい正規表現の構文
入荷番号(123456)	入荷番号\(123456\)
user@example.com	user@example \.com
www.test.abc	www\.test\.abc
file_name.txt	file_name\.txt
価格は10.50	価格は10\.50
C:\Users\Admin	C:\\Users\\Admin
[社外秘]	\[社外秘\]
{請求書}	\{請求書\}
+34 600 123 456	\+34 600 123 456
質問?	質問\
100%保証	100%保証（%はエスケープを必要としない）
アスタリスク*記号	アスタリスク*記号
A|B	A|B
キャレット^start	キャレット\^start
100ドル	ドル\100ドル

正規表現の使用の制限

正規表現は、必要な場合にのみ慎重に使用する必要があります。強力なマッチング機能を備えている一方で、式の設計が不適切であったり過剰であったりすると、メッセージの処理時間が長く

なり、意図しない一致が発生する可能性があります。

注意が必要な特定の構文の1つに、.*があります。これは、「任意の文字、ゼロ回以上」を表します。 式の先頭または末尾に配置すると、過度のバックトラックおよび不要な処理オーバーヘッドが発生する可能性があります。

シスコのドキュメントでは、特定のMIMEパーツと一致する場合、最初または最後に.*を使用するエントリによって、特定の条件下でシステムがロックされる可能性があると説明されています。このため、シスコでは可能な限り先頭または末尾の.*の使用を避けることを推奨しています。

多くのシナリオでは、管理者が単純にinvoiceを記述するだけで、ESAで同じ実用的な結果が得られる場合、.*invoice.*などのパターンを使用します。スキャンエンジンはすでに関連するコンテンツ領域を検索しているため、単語を.*で囲むと冗長性が増し、計算効率が低下することがよくあります。

⚠ 注意：一般的には、正規表現はできるだけ単純かつ正確にしておくことを推奨します。

メッセージフィルタ、コンテンツフィルタ、および辞書

Cisco ESAには、メッセージを評価してアクションを適用するための複数のメカニズムがあります。メッセージフィルタはパイプラインの先頭で動作し、スクリプト形式の構文を使用します。非常に柔軟で、エンベロープデータ、ヘッダー、添付ファイルのプロパティに関する高度なロジックを使用できます。ただし、これらは処理チェーンの早い段階で実行されるため、非効率的なメッセージフィルタはパフォーマンスに悪影響を及ぼす可能性があります。

コンテンツフィルタはグラフィカルインターフェイスを使用して設定され、メッセージが受け入れられると動作します。ほとんどのコンテンツインスペクションの使用例では、コンテンツフィルタはパフォーマンスの観点から見ると管理が容易で安全性が高くなります。

メッセージフィルタとコンテンツフィルタの両方で、正規表現は条件に直接挿入することも、辞書を使用して間接的に挿入することもできます。

辞書を使用すると、管理者は再利用可能な検索用語を一元化できます。各エントリは別個の行に記述され、プレーンテキストまたは正規表現を使用できます。辞書は非ASCII文字もサポートしているため、多言語環境に適しています。

場合によっては、特定の複雑な正規表現の構造が辞書内で同じように動作しないことがあります。この場合、正規表現はディクショナリ内ではなく、フィルタ条件に直接配置する必要があります。

Cisco ESAでは、最大150のコンテンツ辞書を作成できます。デフォルトでは、CLIでdictionaryconfigコマンドを使用して制限を変更しない限り、100のディクショナリを設定できます。

辞書は、用語の重み付けを実装することもできます。各用語には数値の重みを割り当てることが

できます。ESAがメッセージをスキャンすると、その用語の出現数にその重みを掛けます。結果のスコアは、フィルタで定義されたしきい値と比較されます。このスコアモデルにより、より柔軟で段階的なポリシーの適用が可能になります。

さらに、辞書には、社会保障番号や銀行のIDなどの構造化された数値パターンのアルゴリズム検出機能であるスマートIDを含めることができます。

正規表現エンジン

Cisco ESAでは、Python reモジュールスタイルに基づく正規表現を使用します。これは一般的なPython正規表現構文との互換性を提供しますが、完全なPython環境でサポートされているすべての高度な機能がESAで必ずしもサポートされているわけではありません。

正確な文字列照合を行うには、式の先頭に^を、末尾に$を使用してアンカーする必要があります。これらのアンカーがない場合、正規表現エンジンは完全な値ではなく部分文字列を照合できます。

たとえば、次の式を使用します。

```
sun.com
```

次のような文字列を照合します。

```
thegodsunocommando
```

ただし、式は次のようになります。

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

完全に一致する文字列sun.comのみを検索します。

空の文字列を照合する場合は、すべての文字列に効果的に一致するため、「」を使用しないことが重要です。その代わりに、正しい式は次のようになります。

```
^$
```

Cisco ESAはPythonスタイルの正規表現を使用するため、大文字と小文字を区別しない比較を実

行する方法がいくつかあります。

前述のように、デフォルトでは正規表現の大文字と小文字は区別されます。つまり、次の項目を検索します。

```
foo
```

fooにのみ一致し、FOO、Foo、fOoには一致しません。

大文字と小文字を区別しない照合を行う場合は、正規表現の先頭にインラインフラグ(?i)を使用できます。これは、パターンの残りの部分に対して大文字と小文字を無視するように正規表現エンジンに指示します。

例：

```
(?i)foo
```

この式は次と一致します。

- 食べる
- FOO
- Foo
- フロー

大文字小文字を無視して文字列全体を正確に一致させるには、大文字小文字を区別しないフラグをアンカーと組み合わせることができます。

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

これにより、大文字と小文字に関係なく、完全な値が正確に「foo」であることが保証されます。

別の（あまり実用的でない）代替案は、文字クラスを使用してすべての可能な組み合わせを明示的に定義することです。次に例を示します。

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

ただし、この方法は維持が難しくなり、(?i)フラグを代わりに使用できる場合は推奨されません。

ほとんどのESAのシナリオでは、大文字と小文字を区別しない照合を行うために、次の方法が推奨されます。

```
(?i)
```

正規表現の先頭。

非ASCII文字と単語境界

ダブルバイト文字セットを使用する言語では、単語の境界や大文字と小文字の概念が期待どおりに動作しません。\wなどの構文に依存する複雑な式では、エンコードまたはロケールが不明な場合に一貫性のない結果が生成される可能性があります。

このような場合は、辞書設定で単語境界の強制を無効にするか、あいまいな文字クラスへの依存を避けるために式を簡素化することをお勧めします。

ASCII以外の辞書を使用している場合、CLIディスプレイは端末のエンコーディングに応じて文字を正しく表示できません。このような場合は、辞書をテキストファイルにエクスポートし、外部で編集してから、再インポートすることをお勧めします。

効率的なフィルタの記述

特に大量のデータを扱う環境では、フィルタの書き込み時に効率性が重要になります。よく見られる誤りは、類似した一致に対してOR条件の長いチェーンを記述することです。

たとえば、多数の添付ファイル拡張子を個別にチェックすると、正規表現エンジンが繰り返し初期化されます。これにより、CPU使用率が増加し、メンテナンス性が低下します。

多数の個別の比較を記述する代わりに、単一の正規表現の中で別の式を使用してグループ化すると、処理のオーバーヘッドが大幅に削減されます。これにより、正規表現エンジンが呼び出される回数が減り、フィルタの維持が容易になります。

効率的なフィルタ設計は、読みやすさだけでなく、システムのパフォーマンスにも直接影響します。

PDFと正規表現

PDFファイル内のコンテンツが一致すると、PDFの生成方法によっては予期しない結果が生じることがあります。一部のPDFには、内部表現に論理空間や改行が含まれていません。スキャニングエンジンは、ワードの位置決めに基づいて論理的間隔を再構築しようとします。

複数のフォントまたはフォントサイズを使用して単語を作成する場合は、内部表現によってテキストが断片化される可能性があります。たとえば、「callout」という単語は、内部的には「call out」または「c a l lout」と解釈できます。

このような場合、内部表現に正確に連続する文字列が含まれていないため、「callout」という式に一致させようとすると失敗する可能性があります。管理者は、PDF添付ファイルを対象とするコンテンツベースのポリシーを設計する際に、この制限に注意する必要があります。

正規表現のテスト

正規表現を実稼働環境に展開する前にテストすることは、運用上の重要な要件です。構文的に正しいと思われる正規表現は、実際の電子メールトラフィックと比較して評価すると、動作が大きく異なる場合があります。適切なテストを行わないと、フィルタは誤検出を生成したり、目的のパターンを検出できなかったり、パフォーマンスのオーバーヘッドを招いたり、または意図せずに正当なメールフローを中断させたりする可能性があります。

テストは、リスクを最小限に抑えるために、構造化された2段階のプロセスとして、実稼働環境でフィルタを有効にする前に行う必要があります。

フェーズ1：正規表現の設計と検証
最初のフェーズでは、正規表現をCisco ESAに統合する前に、その正規表現の設計と検証に焦点を当てます。

(1)正規表現101等の利用

http://regex101.com（または同等のツール）などのオンラインプラットフォームは、設計フェーズで非常に役立ちます。これらのツールを使用する場合、ESAの正規表現エンジンに近似するようにPythonの種類を選択する必要があります。

これらのプラットフォームを使用すると、管理者は次のことができます。

- 構文が正しいことを検証する
- 特殊文字が正しくエスケープされていることを確認する
- 一致するケースと一致しないケースの両方をテストする
- グループ化と数量化の動作を視覚化
- .*などの貪欲な可能性のある構文を特定する

ただし、これらのツールは標準のPython正規表現の動作をシミュレートし、Cisco ESAに完全には実装されていない機能をサポートできます。したがって、これらのテストは、最終的な互換性テストではなく、事前の検証ツールと見なす必要があります。

2. AIモデルの活用（ChatGPT、コパイロットなど）

AIベースのアシスタントは、特に複雑な照合シナリオの正規表現の作成を高速化できます。希望する動作を自然言語で記述することで、管理者は初期正規表現の提案を取得し、その後で修正することができます。

AIツールは特に以下の点で役立ちます。

- 複雑なグループ化された式の生成
- ビジネス要件の正規表現への変換
- 長いORベースの条件をグループ化された変更に簡素化

しかし、AIで生成された表現は常に批判的に見直されなければなりません。非効率、サポートされていない構造、または非常に複雑なロジックが発生する可能性があります。AI支援は、最終的な検証としてではなく、ドラフト支援として扱う必要があります。AIで生成された式はすべて、構造化された検証方法を使用してテストする必要があります。

フェーズ2:Cisco ESAでのフィルタ動作の検証

式自体の検証が完了したら、2番目のフェーズでは、実際のメッセージ処理に適用した場合のCisco ESA内での動作の確認に重点を置きます。

1. CESコンソールでのトレース機能の使用

Cisco Eメールセキュリティ(CES)コンソールのトレース機能を使用すると、管理者は特定のメッセージの処理方法をシミュレーションおよび分析できます。これは、適用前にフィルタの動作を検証するための最も信頼できる方法の1つです。

トレースは次の情報を提供します。

- メッセージの解析方法
- 評価するフィルタ
- 条件がトリガーされるかどうか
- ルールの実行順序

ESAはMIME解析、ヘッダー正規化、およびコンテンツのデコードを実行するため、アプライアンス内の動作は外部の正規表現テストツールとは異なる場合があります。詳細な手順については、管理者はシスコの公式ドキュメントを参照してください。

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

トレースを使用すると、フィルタが実際の処理エンジン内で期待どおりに動作することを確認できます。

2. ロギング・アクションを使用したフィルタの作成

もう1つの安全で推奨される方法は、メッセージの廃棄、バウンス、または隔離などの積極的なアクションを適用する代わりに、ロギングなどの中断のないアクションを使用してフィルタを展開することです。

一致したエントリを口グに記録するようにフィルタを設定することで、管理者は次のことが可能になります。

- 一致頻度の確認
- 予期しないトリガーの検出
- パフォーマンスへの影響の検証
- 実際のトラフィック動作の分析

このアプローチでは、実稼働トラフィック内の制御されたモニタリングフェーズにフィルタを効果的に配置します。十分な検証が完了し、動作が正しいことが確認されたら、アクションを強制モードに安全に変更できます。

コンテンツフィルタと辞書での式の概要

正規表現が適切に設計および検証されたら、次のステップとして、Cisco ESA内での正規表現の入力方法を理解します。構文は、式がコンテンツフィルタ条件で直接設定されているか、ディクショナリ内で設定されているかによって、若干異なる場合があります。この違いはしばしば混乱を引き起こします。

コンテンツフィルタでの式の概要

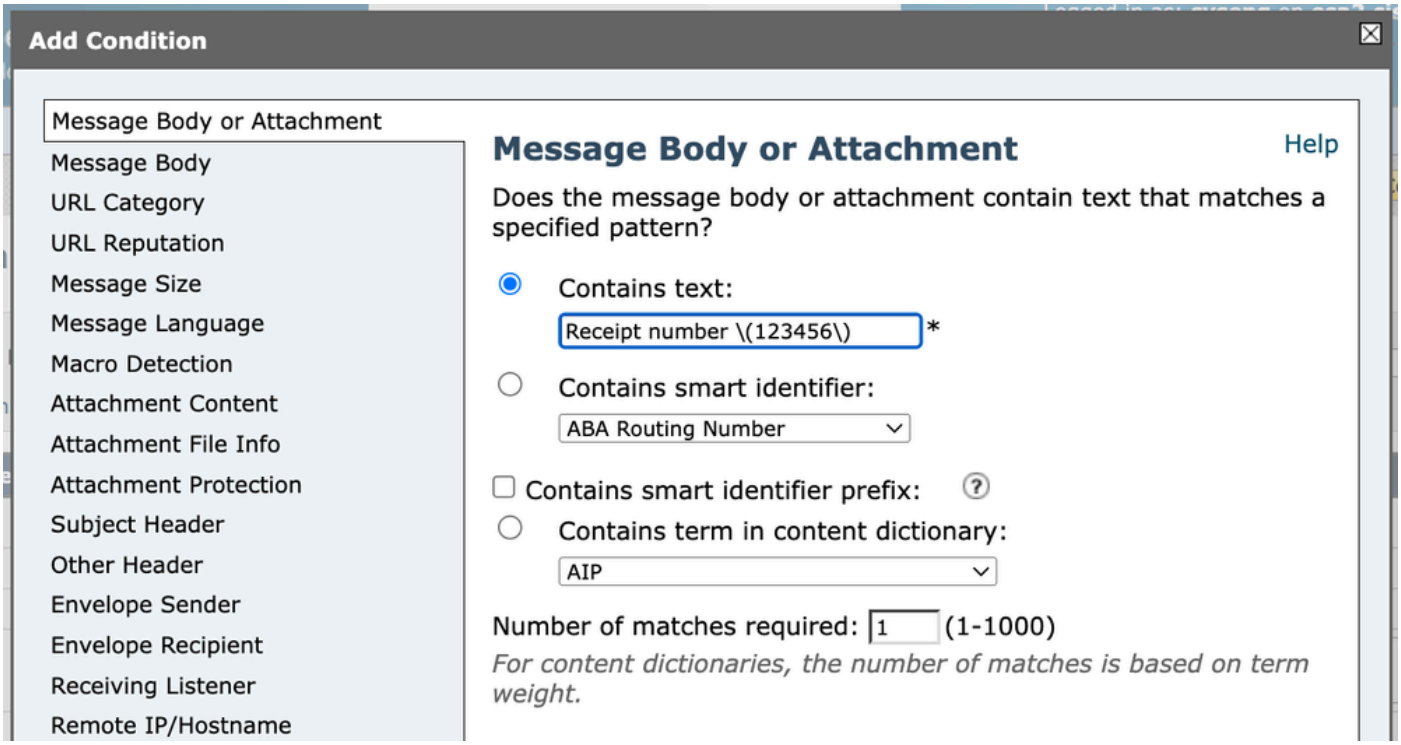
コンテンツフィルタ条件を設定する場合（件名ヘッダーの照合など）、条件フィールドに正規表現を入力する必要があります。リテラルテキストを照合する場合は、次の手順を実行します。

```
Receipt number (123456)
```

カッコは正規表現の特殊文字なので、エスケープする必要があります。

したがって、正規表現は次のように記述する必要があります。

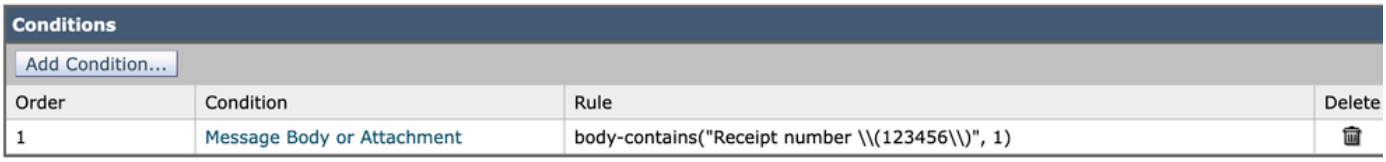
```
Receipt number \(123456\)
```



コンテンツフィルタ1

ただし、GUIまたは高度な設定の出力で完全なフィルタ条件を表示すると、次のように表示されることがあります。

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



コンテンツフィルタ2

これは一見すると混乱を招く可能性があります。ダブルバックスラッシュ(\\)の理由は、バックスラッシュ自体もクォートされた文字列内の特殊文字であるためです。このコンテキストでは、正規表現エンジンのカッコをエスケープするために1つのバックスラッシュが使用され、引用符で囲まれた文字列内のバックスラッシュをエスケープするために2番目のバックスラッシュが使用されます。

実際的には、次のようになります。

\(123456\)は実際の正規表現です。

\\(()は、引用符で囲まれた設定文字列内で\()をシステムが表現する方法です。

表示される場合は異なりますが、評価される論理正規表現は次のとおりです。

入荷番号\(123456\)

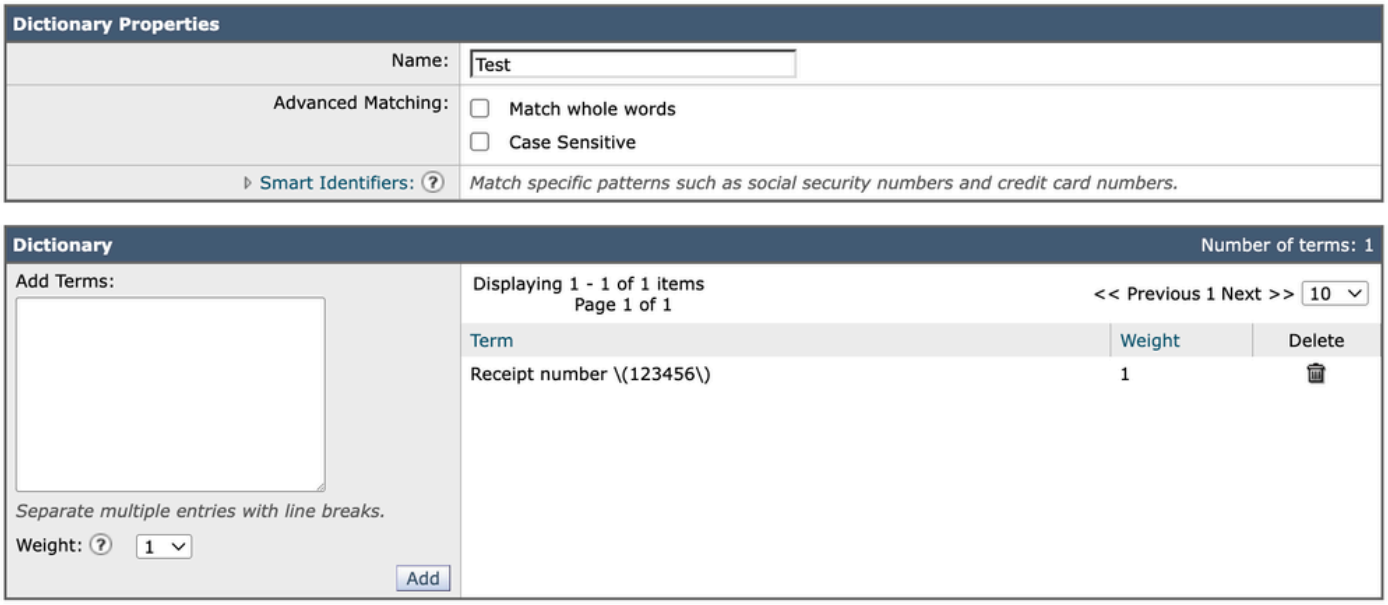
これは、設定出力での文字列エスケープの問題に過ぎません。

辞書での式の紹介

ディクショナリに同じ式を追加すると、エントリは次のように直接挿入されます。

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

この場合、書き込み通りに正確に表示され続けます。コンテンツフィルタGUI表示とは異なり、辞書は視覚的な設定形式でエスケープ層を追加する必要がありません。



辞書

各辞書エントリは、構造に応じてプレーンテキストまたは正規表現として評価されます。特殊文字（この場合はカッコなど）が含まれている場合、式は入力時にすでに正しくエスケープされている必要があります。

「完全に一致する単語」について

辞書を設定する際には、「単語単位で検索」というオプションがあります。 正規表現を使用する場合は、この設定に依存しないことをお勧めします。

その理由は、正規表現アンカーを使用すると、単語境界の動作をより正確に制御できるためです。

例：

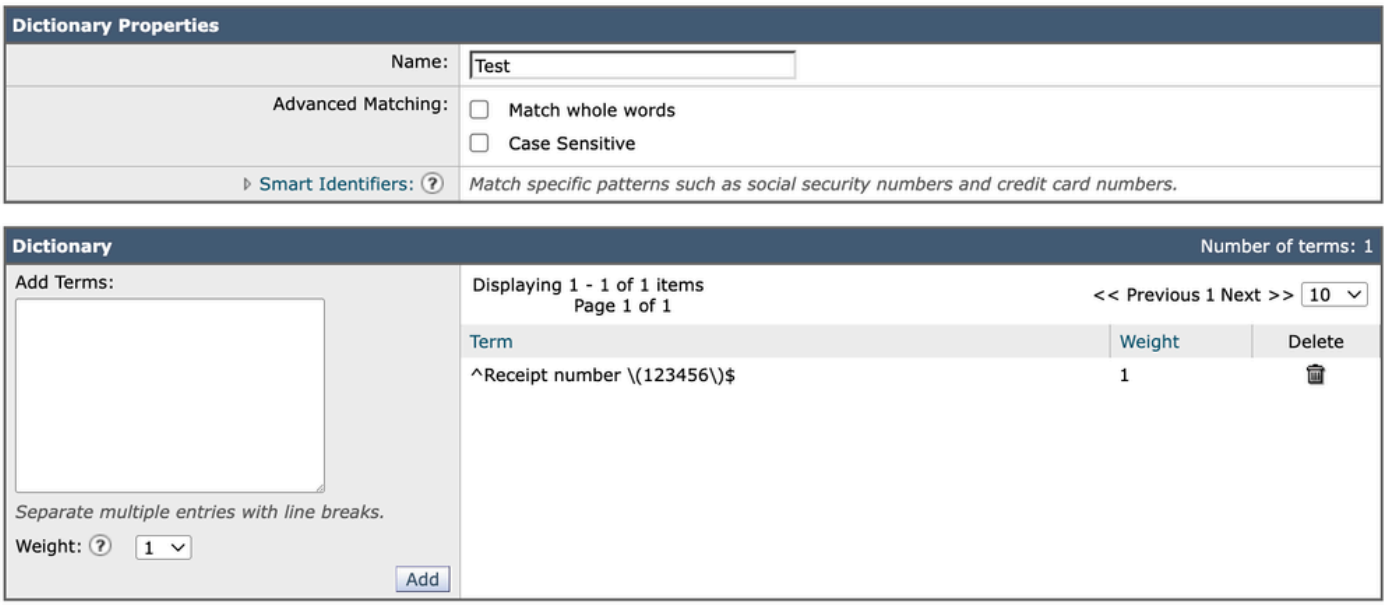
^は、一致が先頭から始まることを保証します。

$を指定すると、照合は末尾で終了します。

次のようなアンカーを使用します。

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

正確な一致動作を明示的かつ予測可能に制御します。このアプローチにより、特に多言語または非ASCII環境において、単語の境界の解釈方法に関する潜在的な曖昧さが回避されます。



辞書2

そのため、「単語単位で検索」オプションを使用するよりも、正規表現で直接一致精度を管理する方が一般的です。

コンテンツフィルタと辞書の微妙な違いを理解することで、式の動作の一貫性が保たれ、実装時の設定ミスのリスクが軽減されます。

Cisco ESAの正規表現コストランキング

Cisco ESAで正規表現を使用して作業する場合、パフォーマンスへの影響は、エンジンがスキャンする必要があるテキストの量と、実行する必要があるバックトラックの量に大きく依存します。ESAはメッセージ本文全体、MIME部分、さらにはデコードされた添付ファイルも評価する必要があるため、非効率なパターンによってCPU使用率が大幅に増加する可能性があります。

計算コストが最も高いものから最も低いものまでの実用的なランクです。

最もコストが高い – リスクが高いパターン

これらの表現は、特に大きなメッセージのパフォーマンスに大きく影響する可能性があります。

ネストされた量指定子（最悪の場合）

例:

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

これらは指数関数的なバックトラッキングシナリオを作成するため、非常に危険です。

別の量子化器の内部に量子化器があると、正規表現エンジンは失敗する前に多くの組み合わせを試みます。

実際のトラフィックでは、これによって重大なCPUスパイクが発生する可能性があります。

推奨事項：境界のない、あいまいなネストされた量指定子を避けてください。

Greedy .*の後に必要なパターン

例：

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

このパターンは、最初にメッセージ全体を消費し、次に必要なサブストリングが見つかるまで文字ごとにバックトラックします。

パターンが存在しない（または最後の近くに表示される）場合、エンジンは必要なトークンを多くの位置でバックトラックしてテストするため、CPUコストが増加します。

ESAでは、本文が大きくなり、MIMEコンテンツが含まれることがあるため、すぐに高額になります。

推奨事項：サブストリングを検出するために。*を先頭に付加しないでください。ESAはすでに評価されたコンテンツを検索し、先頭のワイルドカードはバックトラッキングとCPU使用率を増加させるだけです。

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

共有プレフィクスを持つ大規模な代替

例：

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

複数の代替案が構造を共有する場合、エンジンは各分岐を順に評価します。

早期の分岐がほとんど一致していても遅れて失敗する場合、エンジンは頻繁に再試行します。

これにより、評価時間が大幅に増加します。

中コスト：注意して使用してください

これらのパターンは壊滅的ではありませんが、依然として非効率的である可能性があります。

広範な.*の使用

例：

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

指数関数ではありませんが、.*は無制限の照合を可能にします。予期された部分文字列がすぐに表示されない場合、エンジンはメッセージの大部分をスキャンします。

ESAでは、フィッシングURLの電子メール本文をスキャンするときによく見られます。

遅延修飾子(+?、*?)

例：

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

遅延量子化器は、マッチング戦略を変更します(shortest-first)。 一部のパターンではオーバーマッチングを減らすことができますが、大規模な「検索」ワークロードでは、終了トークンが遅延または欠落している場合に試行を増やすことができます。

多くのESAの使用例では、これらのコマンドは実際の利点をもたらさず、不要な内部再試行を引き起こす可能性があります。

非常に汎用的な文字クラス

例:

```
\S+
.+
```

これらを使用すると、一致範囲を広げて、バックトラッキングパスの数を増やすことができます。

より具体的な文字クラスが常に優先されます。

低コスト：安全で効率的なパターン

これらは、実稼働ESA環境に推奨されます。

固定リテラル

例:

```
text
iw\.adc
```

リテラル文字列は、最も効率的な照合です。エンジンは最小限のオーバーヘッドで簡単な比較を実行します。

アンカーを使用して検索範囲を制限する

特定の位置で一致が予想される場合は、^または$を使用してパターンを固定することを検討してください。アンカーは評価を固定された位置に制限し、エンジンがコンテンツ全体を不必要にスキャンするのを防ぎます。これにより、特に大きなメッセージ本文や構造化されたヘッダーでは、バックトラッキングが減少し、パフォーマンスが向上します。

```
^Invoice$
```

特定の文字クラス

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

これにより、検索領域が大幅に削減され、バックトラッキングが制限されるため、一致する項目が制限されます。

構造化パターンと制約パターン

例：

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

- ドメインは固定されています。
- .*は使用できません。
- 壊滅的なネストパターンを含まない(例： (.*)+)
- 不要な遅延演算子はありません。
- 各断面は拘束されています。

これにより、広範なワイルドカード照合と比較して、CPUへの影響が大幅に軽減されます。

Cisco ESAの実用的なガイダンス

メッセージフィルタまたはコンテンツフィルタの正規表現を設計する場合：

1. パターンが具体的であればあるほど、パフォーマンスは向上します。
2. 本当に必要でない限り。*は避けてください。特に、必要なトークンをその後に置かないでください。
3. 入れ子になった量指定子は使用しないでください。
4. ワイルドカードより明示的な文字クラスを優先します。
5. 適用する前に、必ずモニタモードで新しい式をテストしてください。

正規表現のパフォーマンス比較（Cisco ESAコンテキスト）

パターン	推奨	バックトラッキングリスク	ESAの影響	推奨される代替
https?:\/\/.*/\?text.*	いいえ	高	高い	^https?:\/\/[A-Za-z0-9.-]+(?:\/[^?\s]*)*\/\？テキスト
https?:\/\/.*\？テキスト	⚠注意して	中～高	中～高	^https?:\/\/[^?\s]+\?text$
https?:\/\/.*	いいえ	中～高	中間	^https?:\/\/[A-Za-z0-9.-]+(?:\/[^\s]*)*

.*パスワード	いいえ	高	高い	パスワード$
.*テキスト。*	いいえ	高	高い	テキスト
.*（請求書|支払|振替）	いいえ	高	高い	（請求書|支払|振替）$
(.+)+	never	非常に高い（指数）	深刻	ネストされた量指定子を使用せずに再構築する（例。+）
.*@.*	いいえ	高	高い	[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}
\S+?	理想的でない	中間	中間	\S+または[A-Za-z0-9.-]+のような特定のクラス
.*\/admin	いいえ	高	高い	\/admin$
.*(login|verify).*	いいえ	高	高い	（ログイン|確認）
^.*テキスト	いいえ	高	高い	text$（位置が重要な場合は^text）

結論

正規表現は、Cisco ESA内の強力で柔軟なツールであり、メッセージフィルタとコンテンツフィルタの両方で、正確なコンテンツインスペクションと高度なポリシー適用を可能にします。しかし、その柔軟性が責任となります。不適切に設計された式や十分にテストされていない式は、誤検出、検出の失敗、パフォーマンスの低下、正当なEメールトラフィックの意図しない中断を引き起こす可能性があります。

このため、ESAでの正規表現の使用は、常に構造化された統制のとれたアプローチである必要があります。作成フェーズでは、式が構文的に正しく、適切にエスケープされ、効率的で、意図した目的に論理的に一致していることを確認する必要があります。外部ツールとAIを活用した生成は、このプロセスを大幅に加速できますが、慎重な検証を置き換える必要はありません。

同様に重要なのは、ESA環境内の検証フェーズです。ESAはMIME解析、ヘッダー正規化、およびコンテンツデコードを通じてメッセージを処理するため、実際の動作は理論上の期待とは異なる場合があります。ロギングまたはモニタリングモードで最初にトレースやフィルタの展開などのツールを使用することで、管理者は運用上のリスクを伴わずに正しい動作を確認できます。

要約すると、正規表現はできるだけシンプルに保ち、徹底的にテストし、慎重に配備する必要があります。適切に設計され、適切に検証されたフィルタは、ポリシーを効果的に適用するだけでなく、システムの安定性を保護し、実稼働環境で予測可能な動作を確保します。

文書

Cisco ESA内で正規表現を実装および使用する方法に関する技術的な詳細と公式なガイダンスについては、管理者はシスコの製品ドキュメントを参照する必要があります

「ルール内の正規表現」の項では、規則条件内の構文の考慮事項および使用方法を含め、メッセージフィルタおよびコンテンツフィルタ内で正規表現が評価される方法の概要を説明します。

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

「正規表現の使用に関するガイドライン」の項では、正しい構文、式のアンカー、特殊文字の処理、およびパフォーマンスや照合精度に影響を与える一般的な誤りの回避に関する、実用的な推奨事項について説明しています。

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

正規表現に依存するフィルタを設計またはトラブルシューティングする際は、使用している特定のAsyncOSバージョンに合わせた正式なガイダンスを提供するため、これらの公式リソースを確認することを強くお勧めします。

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。