URLフィルタリングの正規表現のガイドラインとパフォーマンスに関する考慮事項

内容

はじめに

このドキュメントでは、UTDエンジンを使用したURLフィルタリングで正規表現を使用する際のガイドラインとパフォーマンスに関する考慮事項について説明します。UTDエンジンのURLフィルタリングでは、PCRE2の正規表現ライブラリを使用します。

著者：Ciscoエンジニアリング、Eugene Khabarov

前提条件

要 件

次の項目に関する知識があることが推奨されます。

- 正規表現(regex)構文
- URLフィルタリングの概念
- Unified Threat Defense(UTD)設定
- HTTPS/HTTPプロトコルの違い

使用するコンポーネント

このドキュメントの内容は、特定のソフトウェアやハードウェアのバージョンに限定されるものではありません。

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されたものです。このドキュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認してください。

バックグラウンド情報

PCRE2は強力ですが、特定の複雑な式や「greedy」式は過度のバックトラッキングを引き起こし、正規表現エンジンの内部制限に達する可能性があります。これが発生すると、パターンの処理に時間がかかりすぎて、最終的に「一致なし」として処理される可能性があります。

キー ポイント

- PCRE2は、システムリソースを保護するために、バックトラック手順または照合時間に対して内部的な制限を適用します。
- 一部のパターンは構文的に有効ですが、コンピュータ的に安全ではなく、「壊滅的なバックトラック」を引き起こす可能性があります。
- これらの制限を超えると、URLがパターンに論理的に一致する場合でも、正規表現エンジンは処理を中断して一致を返しません。

避けるべきパターン

以下を組み合わせた正規表現の構文は使用しないでください。

- ネストされた量指定子。例：(..+)*、(.*)*、(.+)+
- ワイルドカード(.)を文字列の大部分（特にパターンの終わり近く）に対して繰り返します。
- 繰り返しと共に使用した場合のドメイン名のエスケープされないドット

たとえば、このパターンは構文的には有効ですが、処理にコストがかかる可能性があります。

```
^([a-zA-Z0-9-]+.)*portal.example.com$
```

注：この場合、([a-zA-Z0-9-]+.)*は、ネストされた修飾子(+ inside *)にワイルドカード(.)を加

えたグループです。 一部の非一致入力では、正規表現エンジンが非常に多数のバックトラッキングパスを探索できます。

推奨されるベストプラクティス

ホスト名のドットを常にエスケープする

リテラルドットを照合するには、\.を使用します。次に例を示します。

```
^([a-zA-Z0-9-]+\.)*portal\.example\.com$
```

アンカーパターンと文字の制限

バックトラックを減らすには、^および$を使用し、予期される文字(たとえば、ホストラベルの場合は[a-zA-Z0-9-])に制限します。

可能な限り、ネストされた境界のない繰り返しを避ける

1つの正規表現ですべてをカバーしようとする複雑なパターンよりも、より単純な構造が好まれます。1つの非常に広い式の代わりに、複数の特定のエントリを検討してください。

PCRE2互換テスターでのテストパターン

展開前に、PCRE2対応環境で正規表現パターンをテストし、「壊滅的なバックトラック」または類似の警告を引き起こすパターンを回避します。

注：正規表現パターンがPCRE2エンジンの内部制限に達した場合、URLフィルタリングエンジンでは「一致なし」として処理できます。このような場合、URLの分類は、ホワイトリスト/ブラックリストの正規表現の結果ではなく、カテゴリまたはレピュテーションにフォールバックします。厳密な制限は実装固有であり、リリース間で変更される可能性があります。レジックスは保守的に設計する必要があります。

HTTPとHTTPSのURLマッチングの違い

UTDエンジンは、HTTPSトラフィックとHTTPトラフィックに対して異なる方法でURLを検査します。これは、URLフィルタリング用に正規表現を設計する方法に影響します。

HTTPS(TLS)トラフィック

暗号化されたHTTPSトラフィックの場合、UTDエンジンはデフォルトではペイロードを復号化しません。

- URLフィルタリングでは、Transport Layer Security(TLS)ClientHelloからのサーバ名表示(SNI)を使用します。
- 正規表現パターンはSNIホスト名だけに適用されます(api.example.comなど)。

この場合、ホスト名に基づくパターンはホスト名文字列api.example.comと照合されます。次に例を示します。

```
^([a-zA-Z0-9-]+\.)*example\.com$
```

HTTP（暗号化されていない）トラフィック

プレーンHTTPトラフィックの場合、UTDエンジンは完全なHTTP要求（要求行とヘッダー）を確認できます。

実装によっては、正規表現エンジンに次の文字列を指定できます。

- 完全なURLまたは要求の行(GET /path?param=value HTTP/1.1など)または
- パスと組み合わされたホストヘッダー(例：api.example.com/path)

その結果、HTTPの正規表現の入力には、ベアホスト名だけでなく、/、?、クエリ文字列などの文字を追加できます。

設定の影響

純粋にホスト名用に設計された正規表現(たとえば、api.example.comと一致するもののみ)は、HTTPS(SNI)と正しく一致する可能性がありますが、完全なURLまたはhost+path文字列を含むHTTP要求との一致に失敗します。

同じパターンでHTTPトラフィックとHTTPSトラフィックの両方をフィルタリングするには、次の手順を実行する必要があります。

- 主にホスト名を中心としたパターンの設計
- UTDログでHTTPとHTTPSの両方に対する動作を確認します

確認

デバッグロギングの有効化

ステップ 1：debug utd engine standard url-filtering level infoコマンドを実行して、デバッグロギングを有効にします。

ステップ 2ログを確認するには、show logging process ioxman module utd | include api.example.comコマンドを実行します。

出力例：

```
2025/11/27 11:45:28.195000350 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF event->server_
2025/11/27 11:45:28.195001873 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF   URL: api.exa
2025/11/27 11:45:28.195009216 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF Regex matched
2025/11/27 11:45:28.195022442 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF URLF whitelist
2025/11/27 11:45:33.530605572 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF   URL: api.exa
2025/11/27 11:45:33.530606333 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF Regex not matc
2025/11/27 11:45:33.530614980 {ioxman_R0-0}{255}: [utd] [21292]: (note): :(#0):INSP-URLF URLF whitelist
```

設定例

ホストベースマッチング

example.comのすべてのサブドメインを許可するには、次の推奨されるホスト名優先パターン（ベースライン）を使用します。

```
^([a-zA-Z0-9-]+\.)*example\.com$
```

このパターン：

- example.com、api.example.com、foo.bar.example.comなどに一致します。
- HTTPS(SNI)マッチングに適している
- エンジンで見られる文字列が裸のホスト名の場合もHTTPと一致します。

HTTPホスト/パスマッチング

HTTPにホスト/パスが含まれていて、パスを無視する場合は、ホスト名のプレフィックスを照合し、正規表現を末尾ではなく単語境界で停止できます。*例：

```
^([a-zA-Z0-9-]+\.)*example\.com\b
```

注：ここで、\b（単語の境界）は、明示的な.*ワイルドカードを必要とせずにホスト名を追跡するために/や?などの文字を実際に許可します。これは一般に、末尾に.*を追加するよりも安価であり、境界のないワイルドカードを追加しないようにするため、ガイダンスに合わせて調整してください。

注意:HTTP要求に対して正規表現エンジンに渡される正確な文字列は実装固有であり、進化する可能性があります。疑問がある場合は、ラボ環境でHTTPトラフィックとHTTPSトラフィックの両方に対してパターンをテストし、実稼働環境に展開する前にUTDログで一致を確認してください。

関連情報

- Cisco Catalyst SD-WANセキュリティコンフィギュレーションガイド、Cisco IOS XE Catalyst SD-WANリリース17.x
- シスコのテクニカルサポートとダウンロード

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。