より効率的なメッセージ フィルタを作成するにはどうしますか。

目次

質問

より効率的なメッセージ フィルタを作成するにはどうしますか。

メッセージ フィルターがより長く得ると同時に、ESA の性能特性に影響を与える場合があります。 少数のフィルターか短いフィルターに関しては、効率は重要な問題ではないです。 ただし、より長いフィルターをまたは組み立てた場合実装に多くのフィルターがあれば、ある特定のオペレーションの相対的な効率を意識するはずです。

メッセージをメッセージ パイプラインを通して渡すとき、すべてのメッセージ フィルターは各メッセージに対するアトミック方法で評価される一つの式に結合されます。 これはフィルターの発注が非常に重要である意味し、ことを結合された式のそれ以上の評価をショートできます。 たとえばメッセージに適用するが、1 フィルタは非常に頻繁に適用し、フィルタが可能な限りリストで同様に早く移動する必要があるそれと関連付けられた最終措置 deliver()、bounce()、または drop() がありますいくつかのフィルターがあれば。

ESA が正規表現の処理で非常に効率的であるが、によりかのように追加か不必要な処理を引き起こすために正規表現エンジンを濫用できます。 正規表現の各評価はリソースの同量を大体奪取 します、従ってそれを評価する式の数を減らすことがすばらしい効率にもたらすことをそれは意味します。 たとえば、次のフィルタで、それぞれ評価される添付ファイル名前をドロップ添付ファイルによ名前のパターンに対して比較するとき各「ドロップ添付ファイルによ名前」の正規表現はすべて正規表現評価が 7 回発生することを意味しますで:

```
strip_all_dangerous:  {
i \ \ .pif$;
i \ \ .exe$;
i \ \ .scr$;
i \ \ .msi$;
i \ \ .java$;
i \ \ .dll$;
i \ \ .com$;
}
```

次の例では、結果は同等ですが、例は単一正規表現評価だけ引き起こす多くの効率的によりです:

```
strip_all_dangerous:  {
i \ \PIF|exe|SCR|msi|java|dll|com $;
}
```

第 2 正規表現は最初のフィルタで 7 物より複雑であるが、7 簡単な物より 1 つの複雑な正規表現を評価する多くの効率的です。

ただし、この手法はそのようなフィルタの維持のコストに対して平衡型である必要があります。