# Comment est-ce que j'écris des filtres plus efficaces de message ?

## Contenu



## Question

Comment est-ce que j'écris des filtres plus efficaces de message ?

Pendant que les filtres de message obtiennent plus long, ils peuvent affecter les caractéristiques du fonctionnement de votre ESA. Pour un nombre restreint de filtres ou filtres courts, l'efficacité n'est pas un souci significatif. Cependant, en construisant de plus longs filtres ou si votre implémentation a beaucoup de filtres, vous devriez être conscient de l'efficacité relative de certaines exécutions.

En passant des messages par le pipeline de message, tous les filtres de message sont combinés dans une expression simple qui est évaluée d'une manière atomique contre chaque message. Ceci signifie que la commande des filtres est très importante, et peut se mettre davantage d'évaluation en court-circuit de l'expression combinée. Par exemple, si vous avez un certain nombre de filtres qui s'appliqueront aux messages, mais un filtre s'appliquera très fréquemment et aura un deliver() de mesure finale, un bounce(), ou un drop() associé avec lui, que le filtre devrait être déplacé aussi tôt la liste comme possible.

Bien que l'ESA soit très efficace dans son traitement des expressions régulières, vous pouvez maltraiter l'engine d'expression régulière de façon à entraîner le traitement supplémentaire ou inutile. Chaque évaluation d'une expression régulière prend rudement le même montant de ressources, ainsi il signifie que cela la réduction du nombre d'expressions que vous évaluez rapportera à une plus grande efficacité. Par exemple, dans le filtre suivant, toutes les expressions régulières dans chaque « baisse-connexion-par-nom » sont évaluées individuellement, signifiant que l'évaluation d'expression régulière se produit 7 fois en comparant le nom de connexion contre le modèle dans le baisse-connexion-par-nom :

```
strip_all_dangerous : si (vrai) {
baisse-connexion-par-nom (« (? i) \ \ .pif$") ;
baisse-connexion-par-nom (« (? i) \ \ .exe$") ;
baisse-connexion-par-nom (« (? i) \ \ .scr$") ;
baisse-connexion-par-nom (« (? i) \ \ .msi$") ;
baisse-connexion-par-nom (« (? i) \ \ .java$") ;
baisse-connexion-par-nom (« (? i) \ \ .dll$") ;
baisse-connexion-par-nom (« (? i) \ \ .com$") ;
}
```

Dans l'exemple suivant, les résultats sont équivalents, mais l'exemple est beaucoup plus efficace, entraînant seulement une évaluation simple d'expression régulière :

```
strip_all_dangerous : si (vrai) {
```

```
baisse-connexion-par-nom (« (? i) \ \. (pif|exe|SCR|msi|Javas|DLL|COM) $") ;
}
```

Bien que la deuxième expression régulière soit plus complexe que sept dans le premier filtre, il est beaucoup plus efficace d'évaluer une expression régulière complexe que sept ceux simples.

Cependant, cette technique doit être équilibrée contre le coût de mettre à jour un tel filtre.