# Configuration de la détection précoce des paquets AppID dans Secure Firewall Threat Defense 7.4

## Table des matières


Introduction

Contexte - Problème (Exigences du client)

Nouveautés de Cisco

Présentation des fonctionnalités

Conditions préalables, plates-formes prises en charge, licences

- Plates-formes logicielles et matérielles minimales

Prise en charge de Snort 3, Multi-Instance et HA/Clustering

- Composants utilisés

Détails des fonctionnalités

- Description des fonctionnalités
- Comparaison antérieure à cette version
- Comment ça fonctionne

Workflow API de détection précoce des paquets AppID

- Description des champs API de l'exemple de détecteur personnalisé

Exemple d'utilisation : Comment bloquer le trafic plus rapidement

Procédure pas à pas de Firewall Management Center

- Étapes de création d'un détecteur personnalisé avec l'API

Réinspecter les v/s activés désactivés

Dépannage/Diagnostics

- Présentation des diagnostics
- Localisation du contenu des détecteurs AppID Lua
- Étapes de dépannage

Détails des limitations, problèmes courants et solutions de contournement

- Historique de révision


## Introduction

Ce document décrit comment configurer la détection précoce des paquets AppID dans Cisco Secure Firewall 7.4.

## Contexte - Problème (Exigences du client)

- La détection des applications via Deep Packet Inspection peut nécessiter plusieurs paquets pour identifier le trafic.

- Parfois, lorsque l'adresse IP et/ou le port d'un serveur d'applications sont connus, vous pouvez éviter d'inspecter des paquets supplémentaires.

# Nouveautés de Cisco

- Une nouvelle API Lua AppID basée sur Snort a été créée, qui nous permet de mapper une adresse IP, un port et un protocole aux éléments suivants :
  - Protocole d'application (service appid),
  - Application cliente (appid client) et
  - Application Web (payload appid).
- Les détecteurs d'applications personnalisés peuvent être créés sur FMC à l'aide de cette API pour la détection d'applications.
- Une fois ce détecteur activé, cette nouvelle API nous permettrait d'identifier les applications sur le tout premier paquet d'une session.

# Présentation des fonctionnalités

- L'API est identifiée comme suit :
  - **addHostFirstPktApp** (protocol_appId, client_appId, payload_appId, adresse IP, port, protocole, réinspecter)

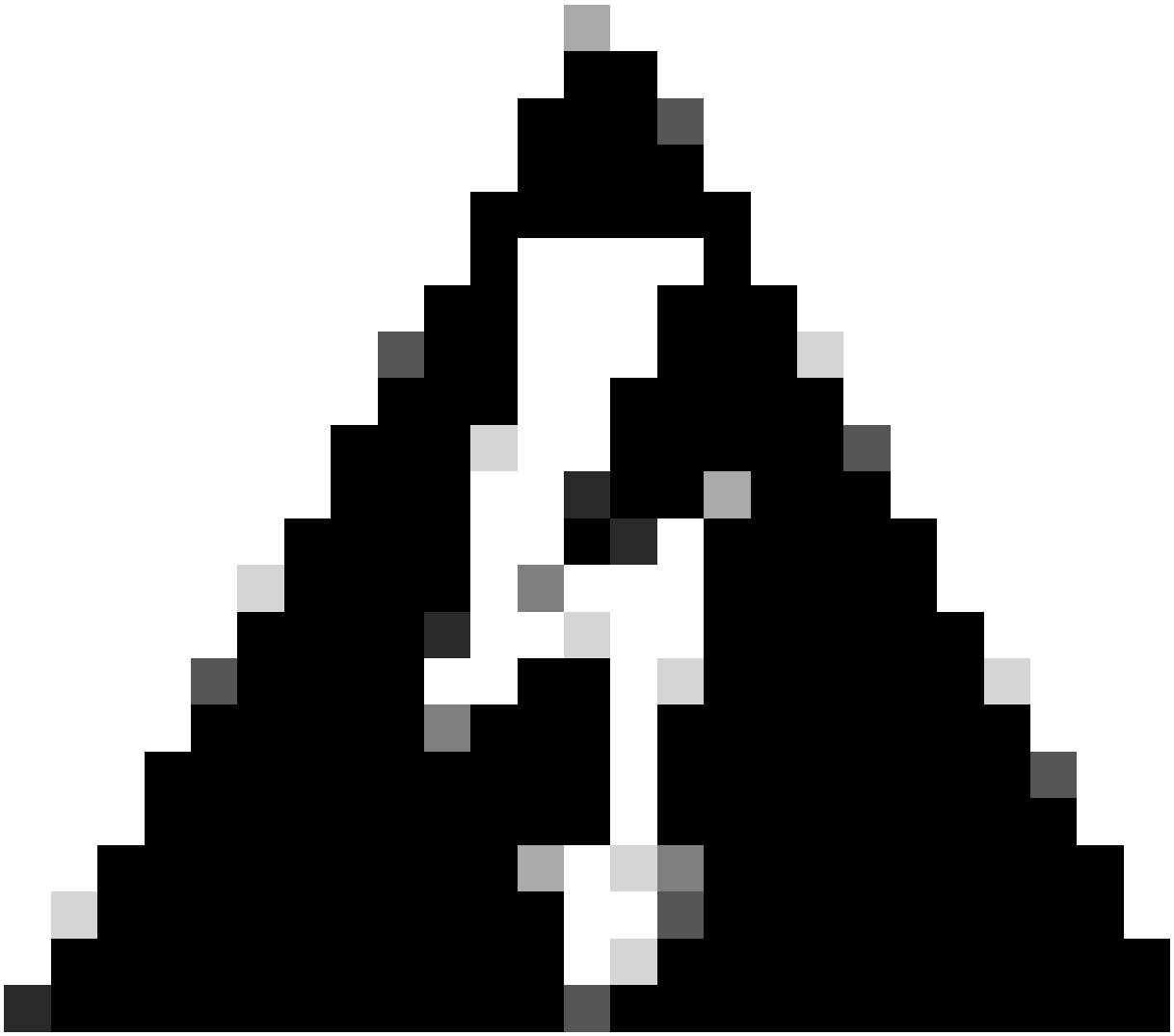
- Une entrée de cache est créée pour chaque mappage créé dans le détecteur d'application personnalisé.
- Le premier paquet de toutes les sessions entrantes est inspecté pour voir si une correspondance est trouvée dans le cache.
- Une fois qu'une correspondance est trouvée, nous attribuons les appids correspondants pour la session et le processus de découverte d'applications s'arrête.
- Les utilisateurs ont la possibilité de réinspecter le trafic même après qu'une correspondance a été trouvée par l'API.
- L'argument reinspect est une valeur booléenne qui indique s'il est nécessaire de réinspecter les applications trouvées sur le premier paquet ou non.
- Lorsque la réinspection est vraie, la détection d'application continue même si l'API trouve une correspondance.
- Dans ce cas, les appids attribués sur le premier paquet peuvent changer.

**Conditions préalables, plates-formes prises en charge, licences**

**Plates-formes logicielles et matérielles minimales**

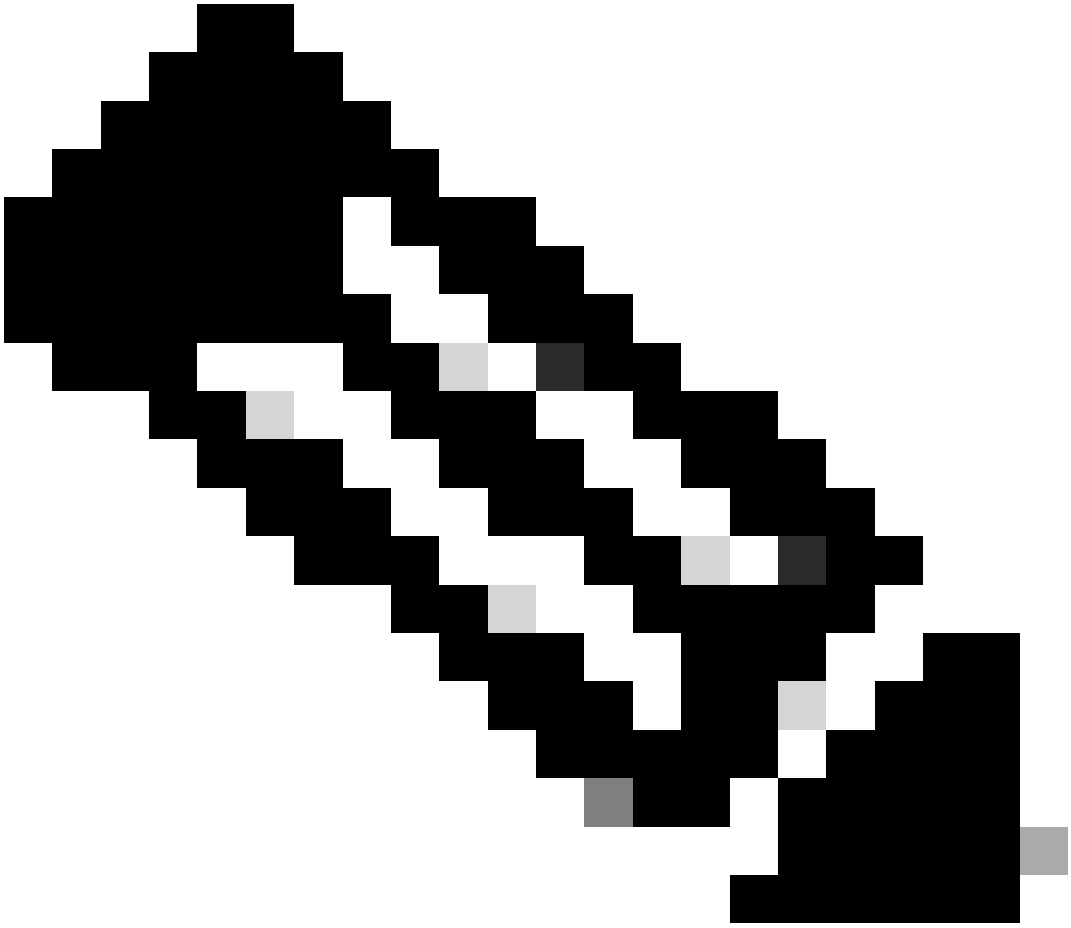
| Application et version minimale | Plate-forme(s) gérée(s) et version prises en charge | Responsable(s) | Remarques |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Pare-feu sécurisé 7.4<br><br>Utilisation de Snort3 | Toutes les plates-formes prenant en charge FTD 7.4 | FMC sur site + FTD | Il s'agit d'une fonctionnalité côté périphérique ; FTD doit être sur 7.4 |



**Avertissement** : Snort 2 ne prend pas en charge cette API.

**Prise en charge de Snort 3, Multi-Instance et HA/Clustering**



**Remarque** : nécessite que Snort 3 soit le moteur de détection.

| FTD | |
|---|---|
| Plusieurs instances prises en charge ? | Oui |
| Prise en charge avec les périphériques haute disponibilité | Oui |

| | |
|---|---|
| Pris en charge avec les périphériques en cluster ? | Oui |

Composants utilisés

Les informations contenues dans ce document sont basées sur les versions de matériel et de logiciel suivantes :
· Cisco Firepower Threat Defense version 7.4 ou ultérieure.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. Si votre réseau est en ligne, assurez-vous de bien comprendre l'incidence possible des commandes.

**Détails des fonctionnalités**

**Description des fonctionnalités**

**Comparaison antérieure à cette version**

| Dans Secure Firewall 7.3 et versions antérieures | Nouveautés de Secure Firewall 7.4 |
|---|---|
| · La détection d'application pour une combinaison IP/Port/Protocole connue n'était disponible que comme option de secours après épuisement de tous les autres mécanismes de détection d'application.<br><br>· Essentiellement, la détection sur le premier paquet d'une session n'était pas prise en charge. | · La nouvelle API du détecteur de lua est évaluée avant tout autre mécanisme de détection d'application,<br><br>· Ainsi, dans la version 7.4, nous prenons en charge la détection sur le tout premier paquet d'une session. |

**Comment ça fonctionne**

- Créer un fichier lua : assurez-vous que le fichier se trouve dans le modèle lua (aucune erreur de syntaxe). Vérifiez également que les arguments fournis à l'API dans le fichier sont corrects.

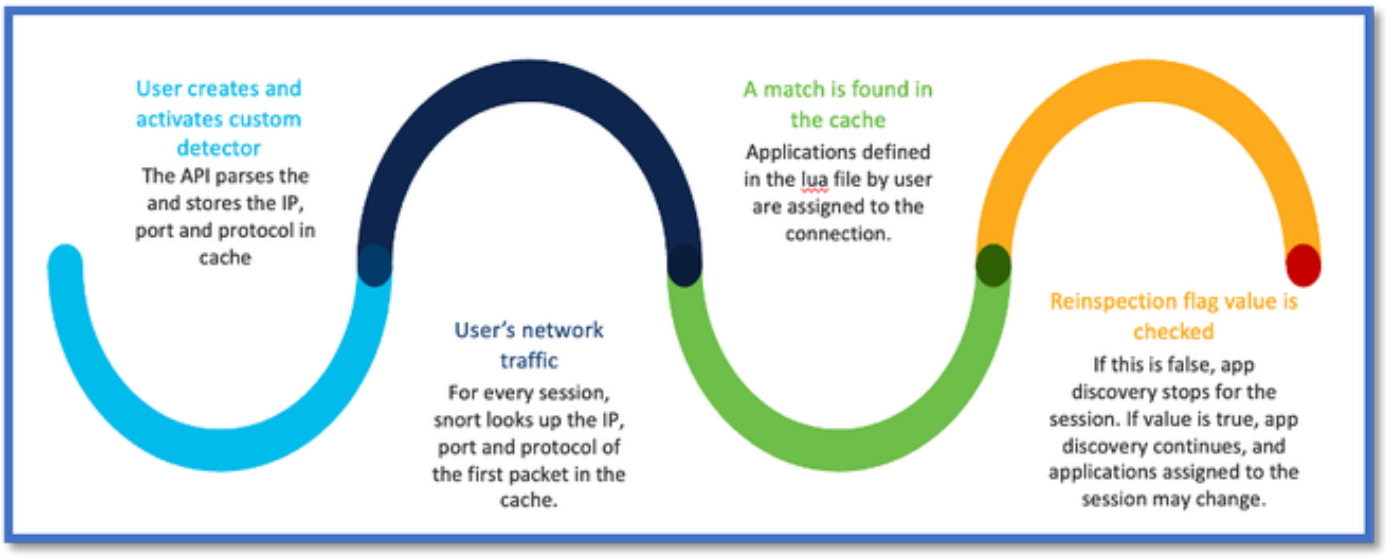
- Créer un nouveau détecteur personnalisé : Créez un nouveau détecteur personnalisé sur FMC et téléchargez votre fichier lua dedans. Activez le détecteur.

- Trafic d'exécution : envoie au périphérique le trafic correspondant à la combinaison IP/port/protocole définie dans le détecteur

d'applications personnalisé.

- Check connection events : sur FMC, vérifiez les événements de connexion filtrés par l'IP et le port. Les applications définies par l'utilisateur seraient identifiées.

**Workflow API de détection précoce des paquets AppID**




**Description des champs API de l'exemple de détecteur personnalisé**

gDetector:addHostFirstPktApp

(gAppIdProto, gAppIdClient, gAppId, 0, "192.0.2.1", 443, DC.ipproto.tcp );

- Les arguments mis en surbrillance sont les valeurs définies par l'utilisateur pour l'indicateur de réinspection, l'adresse IP, le port et le protocole.

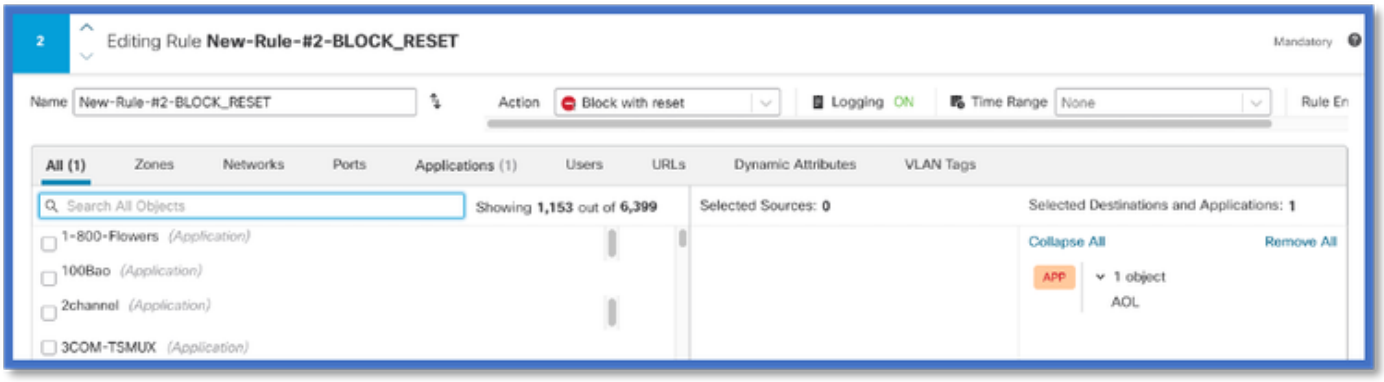
- 0 indique un caractère générique.

| Arguments | Explication | Valeurs attendues |
|---|---|---|
| Indicateur de réinspection | Si un utilisateur préfère inspecter le trafic au lieu d'entreprendre une action de pare-feu basée sur IP/Port/Protocol, il peut activer la valeur d'indicateur de réinspection à 1. | 0 = réinspection désactivée ou<br>1 = réinspection activée |

| | | |
|---|---|---|
| Adresse IP | IP cible (adresse IP unique ou plage d'adresses IP dans un sous-réseau) du serveur. Adresse IP de destination du 1er paquet d'une session. | 192.168.4.198 OU<br>192.168.4.198/24 OU<br>2a03:2880:f103:83:face:b00c:0:25de OU<br>2a03:2880:f103:83:face:b00c:0:25de/32 |
| Port | Port de destination du 1er paquet d'une session. | 0 à 65535 |
| Protocol | Protocole réseau | TCP/UDP/ICMP |

**Exemple d'utilisation : Comment bloquer le trafic plus rapidement**

- Policy View : Règle de blocage pour l'application « AOL ».




- Test du trafic à l'aide de curl avec : curl https://www.example.com v/s curl https://192.0.2.1/ (une des adresses IP de TEST)

**<#root>**

```
> curl https://www.example.com/
```

```
curl: (35) OpenSSL SSL_connect: SSL_ERROR_SYSCALL in connection to www.example.com:443
```

```
> curl https://192.0.2.1/
```

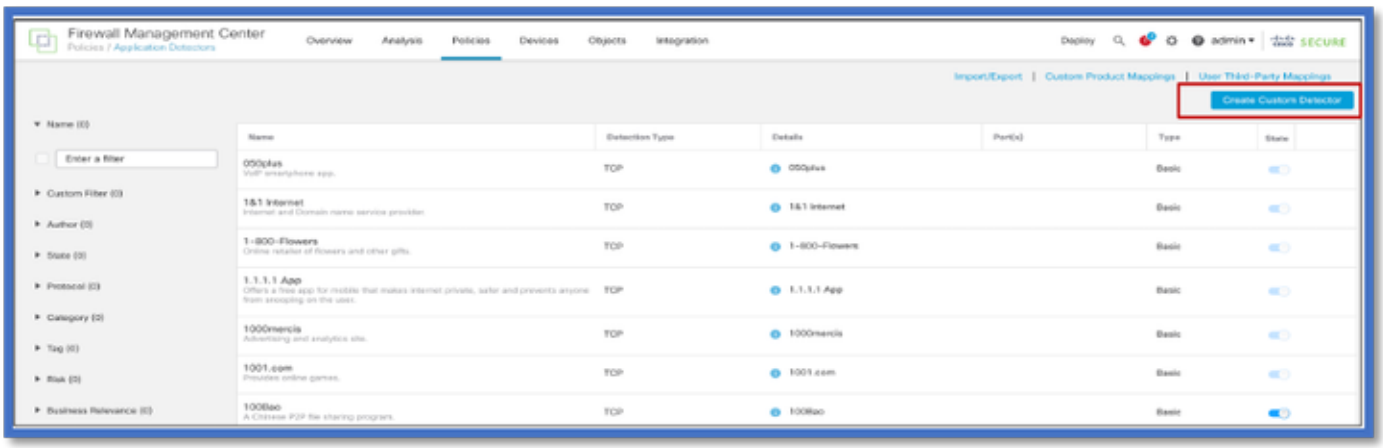
```
curl: (7) Failed to connect to 192.0.2.1 port 443: Connection refused
```

# Procédure pas à pas de Firewall Management Center

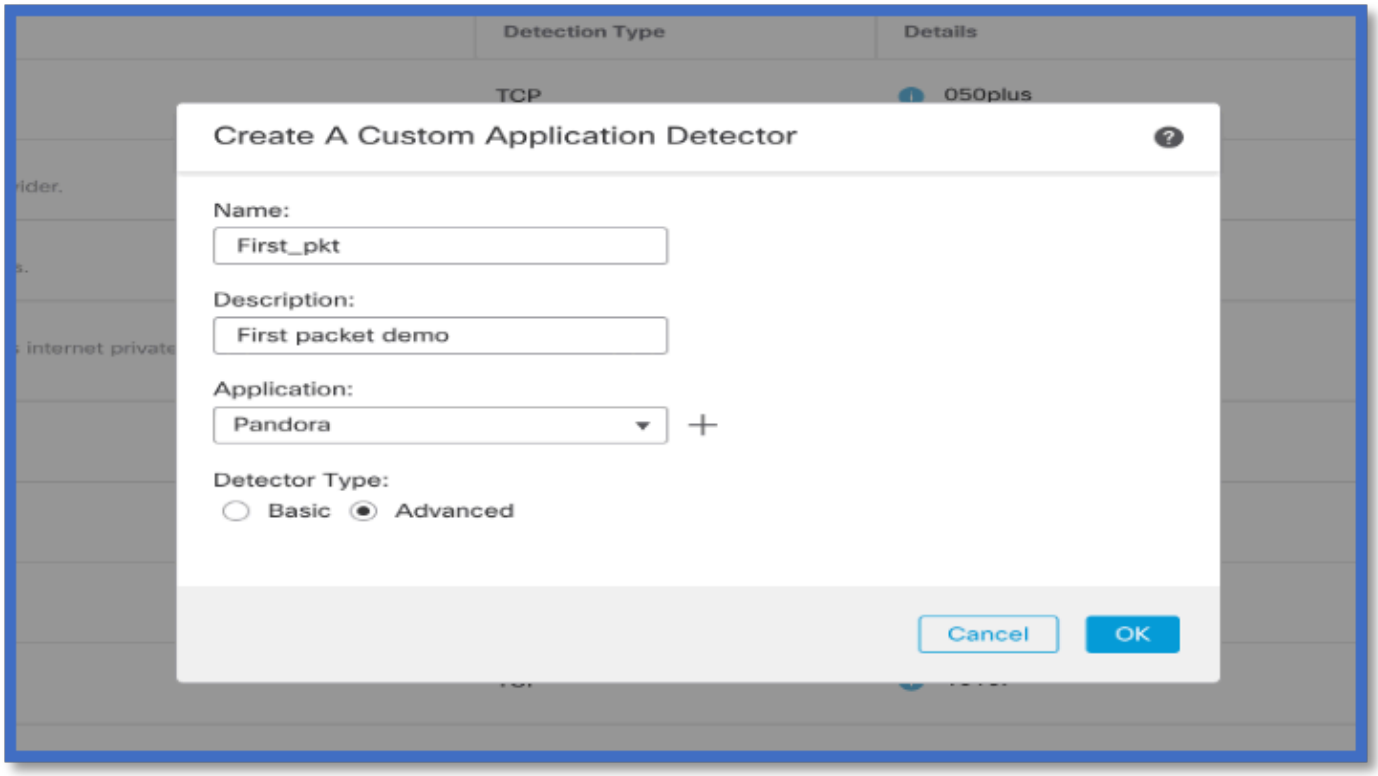
## Étapes de création d'un détecteur personnalisé avec l'API

Créez un nouveau détecteur personnalisé sur le FMC à partir de :

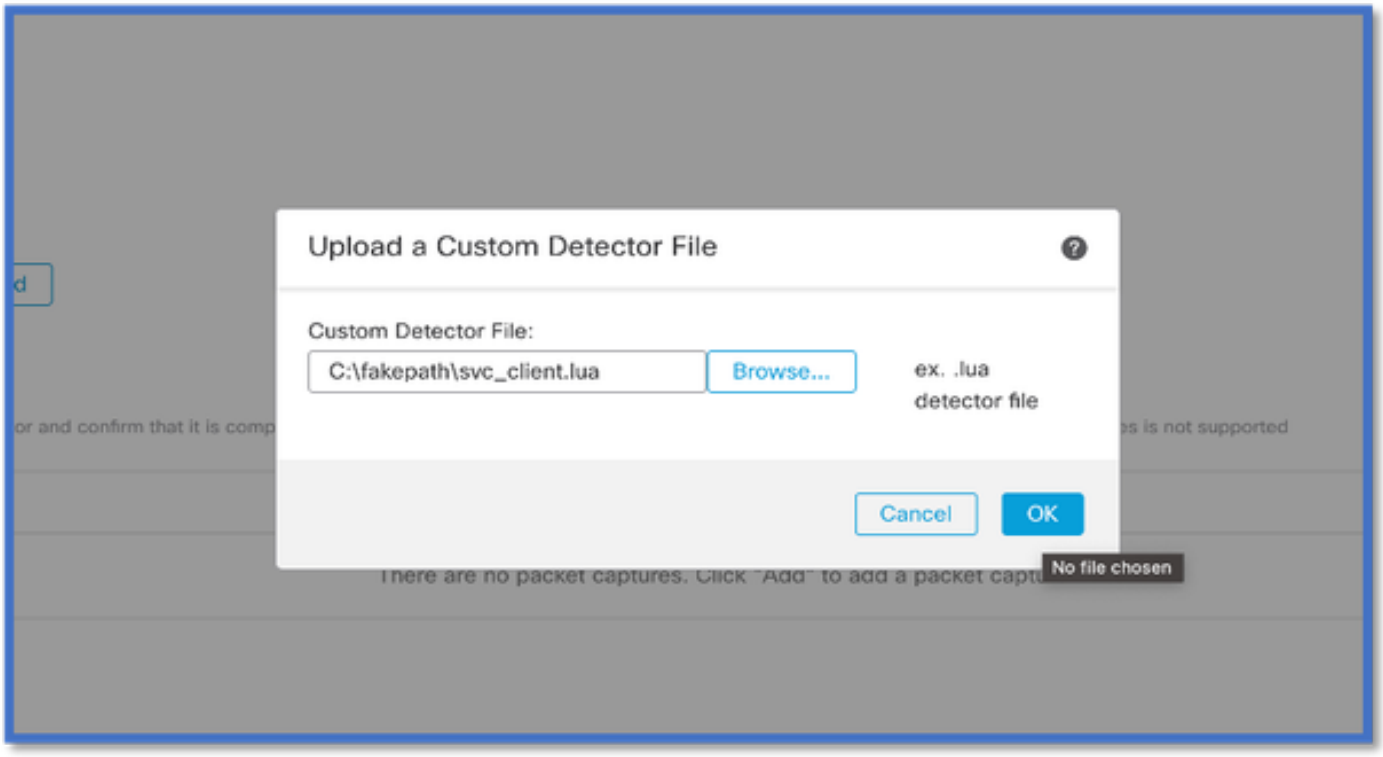
- Policies > Application Detectors > Create Custom Detector .



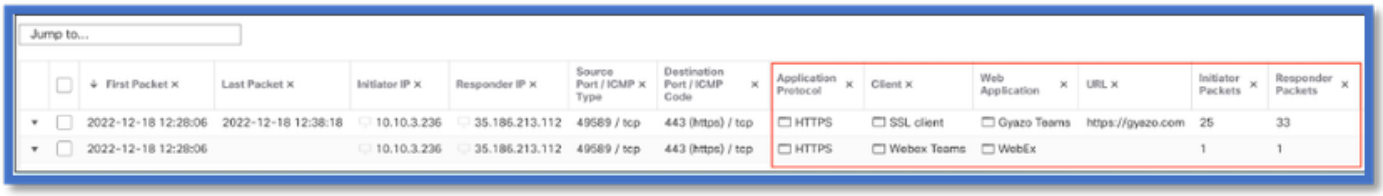

- Définissez le nom et la description.
  - Sélectionnez l'application dans le menu déroulant.
  - Sélectionnez Type de détecteur avancé.



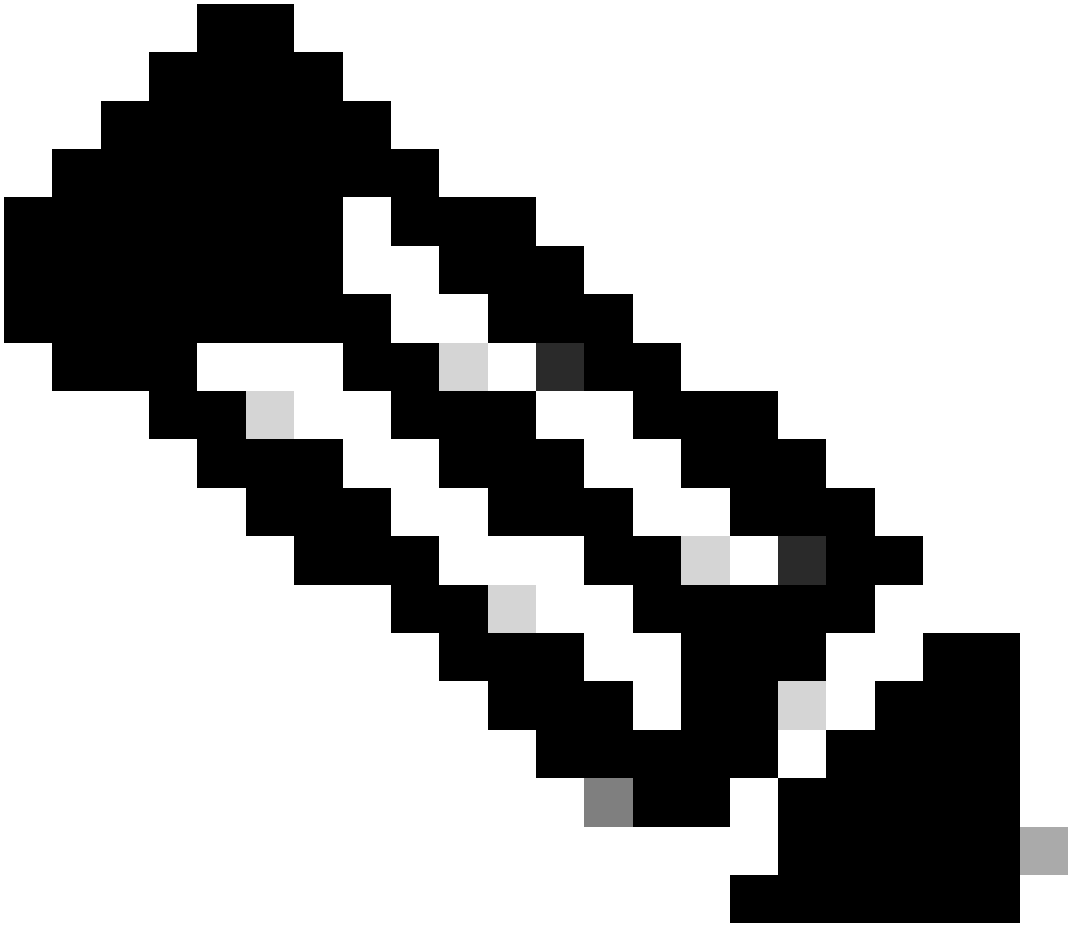

- Téléchargez le fichier Lua sous Critères de détection. Enregistrez et activez le détecteur.




**Réinspecter les v/s activés désactivés**




- Les deux événements indiquent le début de la connexion par rapport à la fin de la connexion lorsque la réinspection est activée.



**Remarque** : éléments à noter :

1. Les « équipes HTTPS, Webex et Webex » sont identifiées par l'API au début de la connexion. Comme la réinspection est vraie, la découverte des applications se poursuit et les identifiants des applications sont mis à jour en « HTTPS, SSL Client et Gyazo Teams ».

2. Notez le nombre de paquets initiateur et répondeur. Les méthodes de détection d'applications régulières nécessitent beaucoup plus de paquets que l'API.

**Dépannage/Diagnostics**

**Présentation des diagnostics**

- De nouveaux journaux sont ajoutés dans le débogage d'identification d'application de prise en charge du système pour indiquer si des applications sont trouvées par la 1ère API de détection de paquet.
- Les journaux indiquent également si l'utilisateur a choisi de réinspecter le trafic.
- Le contenu du fichier du détecteur de lua téléchargé par l'utilisateur se trouve sur le FTD sous /var/sf/appid/custom/lua/<UUID> .
- Toutes les erreurs du fichier lua sont enregistrées dans le fichier FTD du fichier /var/log/messages au moment de l'activation du détecteur.

CLI : prise en charge du système application-identification-debug

<#root>

192.0.2.1 443 -> 192.168.1.16 51251 6 AS=4 ID=0 New AppId session

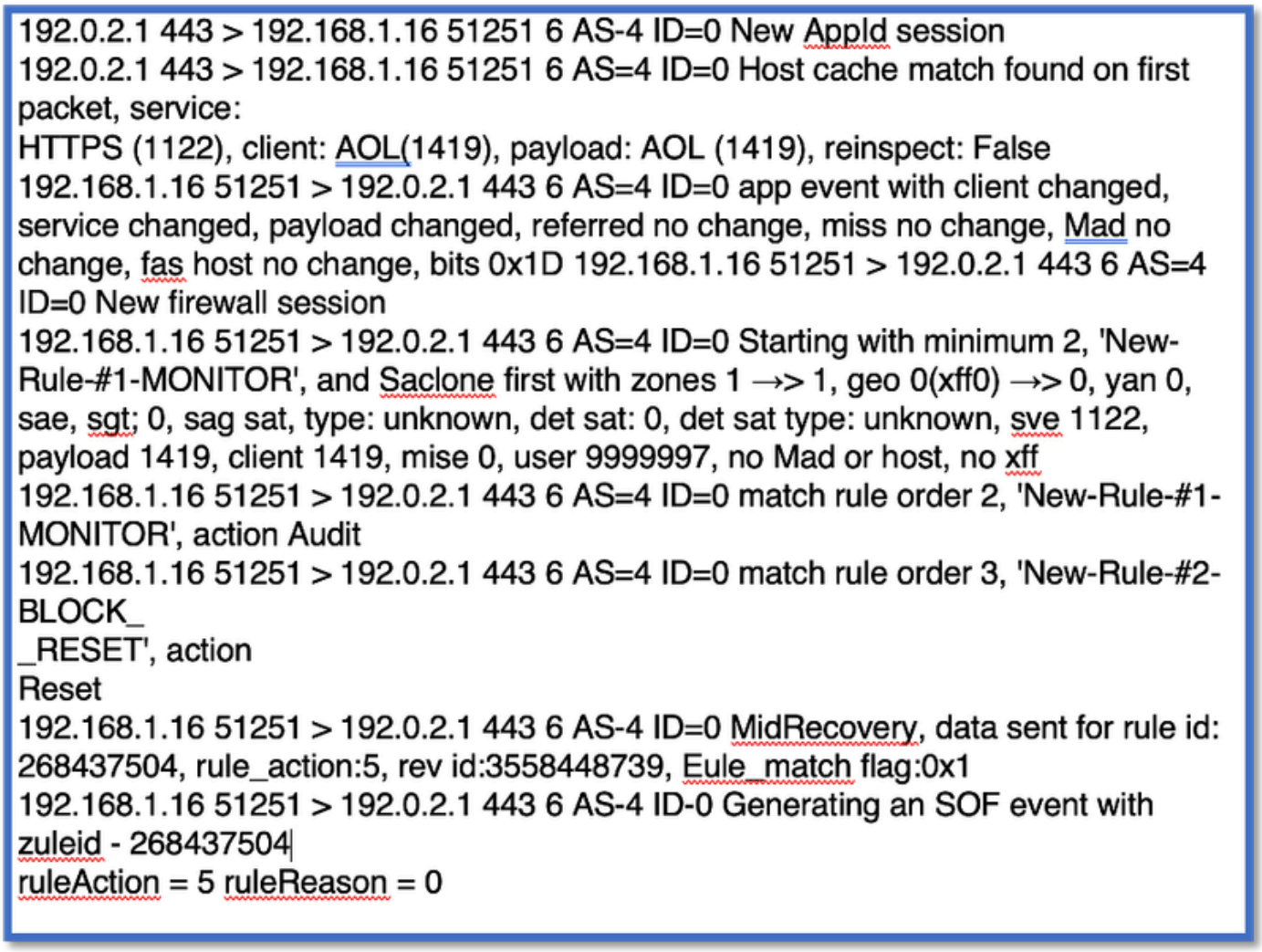
**192.0.2.1 443 -> 192.168.1.16 51251 6 AS=4 ID=0 Host cache match found on first packet, service: HTTPS(1**

192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 app event with client changed, service changed, payload
192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 New firewall session
192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 Starting with minimum 2, 'New-Rule-#1-MONITOR', and Src
192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 match rule order 2, 'New-Rule-#1-MONITOR', action Audit

**192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 match rule order 3, 'New-Rule-#2-BLOCK_RESET', action Re**

192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 MidRecovery data sent for rule id: 268437504, rule_acti
192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 Generating an SOF event with rule_id = 268437504 ruleAc

**192.168.1.16 51251 -> 192.0.2.1 443 6 AS=4 ID=0 reset action**




**Localisation du contenu des détecteurs AppID Lua**

Pour vérifier si le détecteur Lua avec cette nouvelle API existe sur le périphérique/FTD, vous pouvez vérifier si l'API addHostFirstPktApp est utilisée dans les 2 dossiers de détecteur d'application :

1. Détecteurs AppID VDB -/var/sf/appid/odp/lua

2. Détecteurs personnalisés -/var/sf/appid/custom/lua

Par exemple :grep addHostFirstPktApp *  dans chaque dossier.

Exemples de problèmes :

- Problème : Le détecteur de Lua personnalisé n'est pas activé sur FMC.

Emplacement à vérifier :  /var/sf/appid/custom/lua/

Résultat attendu : Un fichier pour chaque détecteur d'application personnalisé activé sur le FMC doit exister ici. Vérifiez que le contenu correspond au fichier lua téléchargé.

- Problème : le fichier du détecteur de lua téléchargé comporte des erreurs.

Fichier à vérifier : /var/log/messages on FTD

Journal des erreurs :

<#root>

Dec 18 14:17:49 intel-x86-64 SF-IMS[15741]:

**Error - appid: can not set env of Lua detector /ngfw/var/sf/appid/custom/lua/6698fbd6-7ede-11ed-972c-d12**

**Étapes de dépannage**

Problème : les applications ne sont pas correctement identifiées pour le trafic acheminé vers l'adresse IP et le port définis par l'utilisateur.

Étapes de dépannage :

- Vérifiez que le détecteur de lua est correctement défini et activé sur le FTD.
  - Vérifiez le contenu du fichier lua sur le FTD et assurez-vous qu'aucune erreur ne s'affiche lors de l'activation.
- Vérifiez l'adresse IP de destination, le port et le protocole du premier paquet de la session de trafic.
  - Il peut correspondre aux valeurs définies dans le détecteur de lua.
- Vérifiez la commande system-support-application-identification-debug.

- Recherchez la ligne Host cache match found on first packet. Si elle est manquante, cela indique qu'aucune correspondance n'a été trouvée par l'API.

**Détails des limitations, problèmes courants et solutions de contournement**

Dans la version 7.4, il n'y a pas d'interface utilisateur pour utiliser l'API. La prise en charge de l'interface utilisateur sera ajoutée dans les versions futures.

Historique de révision

| Révision | Date de publication | Commentaires |
|---|---|---|
| 1.0 | 18-juil-2024 | Première publication |

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.