Référence de l'inspecteur Snort 3

Dernière modification : 2025-04-30

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

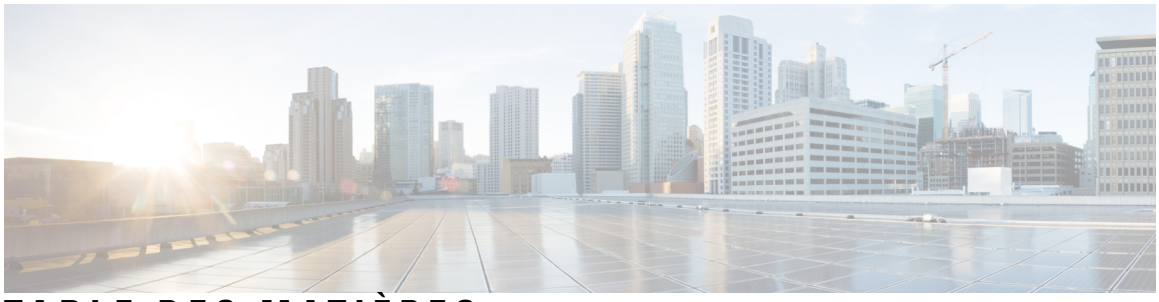
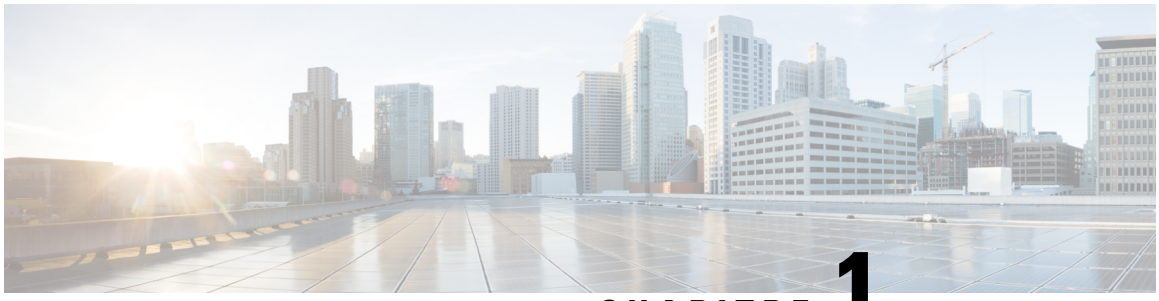


TABLE DES MATIÈRES



CHAPITRE 1

Introduction

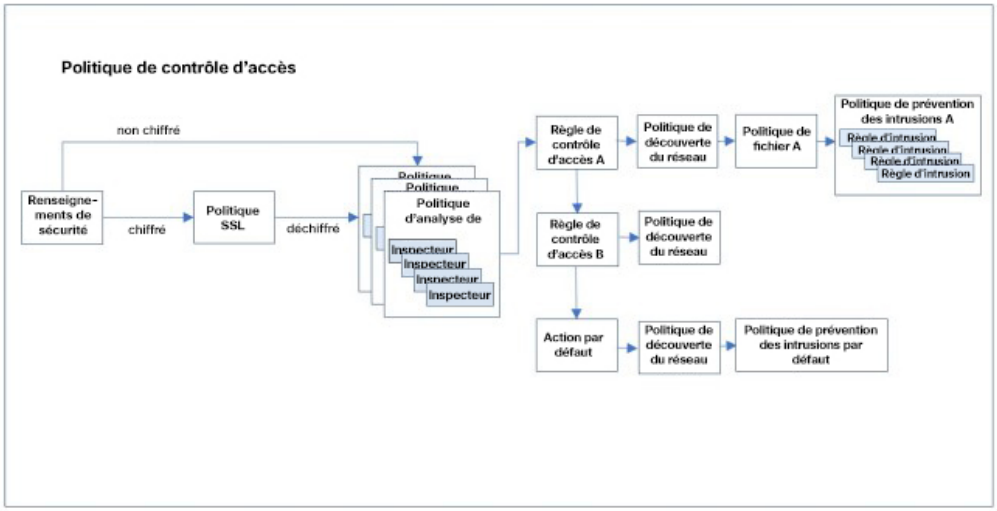
À propos de l'inspection Snort 3

Le système de prévention des intrusions Snort analyse le trafic réseau en temps réel pour assurer une inspection approfondie des paquets. Snort peut détecter et bloquer les anomalies au sein du trafic, ainsi que les sondes et les attaques de réseau. Snort 3 est la dernière version de Snort. Pour en savoir plus, consultez https://snort.org/snort3.

Snort a été conçu pour offrir de hautes performances et une grande évolutivité. Snort comprend un ensemble de modules d'extension configurables appelés *inspecteurs*. Un inspecteur Snort peut détecter et analyser le trafic pour un certain type de protocole réseau ou de sonde, normaliser les messages afin d'améliorer l'analyse des paquets et inspecter des types spécifiques de fichiers intégrés dans un message. Vous configurez les inspecteurs Snort dans une politique d'analyse de réseau (NAP) et activez les règles de prévention des intrusions dans une politique de prévention des intrusions.

Stratégies de contrôle d'accès

Les stratégies de contrôle d'accès traitent le trafic en plusieurs étapes. Le diagramme suivant présente un exemple de déploiement de stratégie. Les éléments abordés dans ce document sont les inspecteurs Snort 3 ainsi que les options de règles utilisées dans les règles de prévention des intrusions, tous en bleu.



Les politiques d'analyse de réseau vous permettent de configurer des inspecteurs Snort 3 pour déterminer le protocole du trafic, et pour extraire et normaliser les données. Vous pouvez configurer différentes politiques d'analyse de réseau, chacune utilisant un ensemble d'inspecteurs Snort 3 configuré de manière unique pour normaliser les données. Les inspecteurs peuvent générer une alerte lorsqu'ils détectent des anomalies dans le flux de données, mais leur objectif principal est de préparer les données pour les règles de prévention des intrusions. Les politiques de prévention des intrusions appliquent leurs règles configurées pour examiner les données et y déceler des signes d'évasion, d'intrusion ou d'attaque.

Dans le cadre d'une politique d'analyse de réseau, vous pouvez personnaliser le comportement d'inspection des données à l'aide d'un protocole particulier en définissant des paramètres de configuration spécifiques à l'inspecteur qui gère ce protocole. Par exemple, pour configurer le comportement d'inspection des données POP, définissez les paramètres de configuration de l'inspecteur `pop`.

Vous pouvez également personnaliser la politique de prévention des intrusions associée à certains protocoles en créant des règles de prévention des intrusions personnalisées à l'aide d'options de règles spécifiques à ces protocoles.

Si vous établissez une configuration complexe en utilisant plusieurs politiques d'analyse de réseau et plusieurs politiques de prévention des intrusions, le système choisit d'abord la politique d'analyse de réseau pour gérer les données. Une fois que la politique d'analyse de réseau a appliqué les inspecteurs appropriés pour son analyse, les données ne sont pas automatiquement transférées à la politique de prévention des intrusions associée à ce protocole. La stratégie de contrôle d'accès effectue des tests supplémentaires pour déterminer quelle politique de prévention des intrusions obtient les données. Par conséquent, lors de la configuration de vos stratégies de contrôle d'accès, de vos politiques d'analyse de réseau et de vos politiques de prévention des intrusions, vous devez vous assurer que les données sont analysées par la bonne paire de politiques d'analyse de réseau et de prévention des intrusions. Pour obtenir plus d'informations, reportez-vous au Guide de configuration Cisco Secure Firewall Management Center pour Snort 3.

Mises à jour des règles de prévention des intrusions

Cisco publie régulièrement des mises à jour des règles de prévention des intrusions sous la forme de LSP (Lightweight Security Packages). Ces mises à jour peuvent modifier les valeurs par défaut des options de règles de prévention des intrusions et des paramètres de configuration d'un inspecteur Snort 3.

Configuration de l'inspecteur

Vous pouvez activer et désactiver les inspecteurs Snort, ainsi qu'afficher et modifier leurs configurations par le biais de l'interface Web du Cisco Secure Firewall Management Center. L'interface Web du Cisco Secure Firewall Management Center utilise le format JSON pour décrire les configurations de l'inspecteur. Pour obtenir plus d'informations, reportez-vous au Guide de configuration Cisco Secure Firewall Management Center pour Snort 3.

Pour utiliser un inspecteur, vous devez l'activer par le biais de l'interface Web du centre de gestion. En outre, pour les inspecteurs de service, vous devez configurer une entrée dans l'inspecteur `binder`. Pour en savoir plus, consultez Présentation de l'inspecteur de binder, à la page 15.

Le guide de référence des inspecteurs Snort 3 présente les configurations par défaut des paramètres des inspecteurs Snort 3 et les options de règles de prévention des intrusions intégrées. Votre système peut utiliser des valeurs par défaut différentes en fonction des mises à jour des LSP ou des politiques d'accès au réseau de base fournies avec le système. Pour mieux comprendre les paramètres d'inspecteur de vos politiques d'accès au réseau, consultez les paramètres dans l'interface Web centre de gestion.

Présentation des inspecteurs Snort 3

Les inspecteurs Snort 3 sont des modules d'extension qui analysent et normalisent les paquets, comme les préprocesseurs Snort 2. La liste des inspecteurs et des paramètres Snort 3 est différente de celle des préprocesseurs et des paramètres Snort 2.

Inspecteurs Snort 3

- Inspecteur d'analyse de ports, à la page 145
- Filtre de débit, à la page 161
- Inspecteur S7CommPlus, à la page 167
- Inspecteur SIP, à la page 171
- Inspecteur SMTP, à la page 179
- Inspecteur SSH, à la page 193
- Inspecteur de flux ICMP, à la page 197
- Inspecteur de flux IP, à la page 199
- Inspecteur de flux TCP, à la page 203
- Inspecteur de flux UDP, à la page 215
- Inspecteur Telnet, à la page 217

Pour chaque inspecteur Snort 3, ce document décrit ce qui suit :

- Informations générales sur le rôle et les fonctionnalités de l'inspecteur
- Type d'inspecteur :
  - Service : inspecteur qui analyse les unités de données de protocole (PDU) utilisées dans les protocoles de service Internet (HTTP, FTP, TCP ou UDP). Par exemple : `http_inspect`, `ftp_server`.
  - Passif : inspecteur qui fournit uniquement la configuration (`ftp_client`, `ftp_server`) ou qui facilite d'autres processus (`binder`).
  - Paquet : inspecteur qui analyse les paquets bruts en amont du traitement effectué par un autre inspecteur. Par exemple : `normalizer`.
  - Sonde : inspecteur qui traite tous les paquets une fois que toutes les détections sont terminées. Par exemple : `port_scan`.
  - Flux : inspecteur qui effectue le suivi du flux, la défragmentation du protocole Internet et le réassemblage TCP. Par exemple : `stream_tcp`, `stream_ip`.
  - Module de base : composant configurable et intégré de Snort 3 qui fournit des fonctionnalités permettant de prendre en charge le processus d'inspection de plusieurs types de trafic. Par exemple : `rate_filter`.
- Utilisation :
  - Inspect : inspecteur qui doit être configuré dans une politique d'analyse de réseau (NAP). Par exemple : `imap`, `ssh`.
  - Global, Contexte : inspecteurs qui ne doivent être configurés qu'une seule fois. Par exemple : `port_scan, rate_filter`.
- Type d'instance :
  - Singleton : inspecteur qui doit être configuré pour une seule instance dans une politique d'accès au réseau. Pour en savoir plus, consultez Inspecteurs Singleton, à la page 5.

- Multiton : inspecteur qui doit être configuré pour plusieurs instances dans une politique d'accès au réseau (NAP). Une NAP peut contenir plusieurs instances différenciées par réseau, port ou VLAN. Chaque instance est configurée individuellement pour traiter un segment de trafic spécifique. Pour en savoir plus, consultez Inspecteurs Multiton, à la page 6.

- Autres inspecteurs requis : de nombreux inspecteurs dépendent d'autres inspecteurs pour traiter entièrement le flux de données. Lorsqu'un inspecteur nécessite la configuration d'autres inspecteurs, la documentation identifie ces derniers.
- Bonnes pratiques en matière de configuration de l'inspecteur : recommandations visant à optimiser les performances de chaque inspecteur.
- Paramètres de configuration de l'inspecteur : vous pouvez définir les paramètres de configuration dans l'interface Web du centre de gestion sous **Policies (politiques) > Access Control (contrôle d'accès) > Network Analysis Policy (politique d'analyse de réseau) >** ***Policy Name (nom de la politique)*** **> Snort 3 Version (version de Snort 3) >** ***Inspector Name (nom de l'inspecteur)***.

Remarque Avant de modifier les paramètres d'un inspecteur, nous vous recommandons de vous familiariser avec l'interaction entre l'inspecteur et les règles de prévention des intrusions activées.

- Règles : les inspecteurs Snort 3 utilisent des règles pour générer des événements. Les règles intégrées peuvent contenir des types de classes, des références et d'autres métadonnées.
- Options des règles de prévention des intrusions : personnalisez les règles de prévention des intrusions en définissant les options correspondantes en fonction du type de données gérées par l'inspecteur. Pour plus de renseignements sur la gestion des règles de prévention des intrusions personnalisées, consultez le Guide de configuration Cisco Secure Firewall Management Center pour Snort 3.

Remarque La création de règles de prévention des intrusions personnalisées est une tâche complexe qui requiert une attention particulière. Vous devrez peut-être utiliser des inspecteurs et des options de règles qui ne sont pas décrits dans cette documentation. Pour utiliser certains des inspecteurs et options de règles de prévention des intrusions présentés dans ce document, vous devrez configurer des paramètres spécifiques décrits dans la documentation ouverte (Open Source) de Snort. Certaines options de règles ont une incidence sur la fonctionnalité de recherche de modèle rapide de Snort ou sur l'emplacement du curseur de détection. Pour plus de renseignements, consultez la documentation ouverte (Open Source) de Snort 3, disponible à l'adresse https://www.snort.org/snort3.

Inspecteurs Singleton

Une politique d'accès au réseau (NAP) ne peut utiliser qu'une seule instance d'un inspecteur singleton.

- Contrairement à un inspecteur multiton, un inspecteur singleton ne peut prendre en charge qu'une seule instance par NAP.
- Un inspecteur singleton peut ne pas s'appliquer à certains flux spécifiques.

Par exemple :

```
{
   "normalizer":{
      "enabled":true,
      "type":"singleton",
      "data":{
         "ip4":{
            "df":true
         }
      }
   }
}
```

Inspecteurs Multiton

Une politique d'accès au réseau peut utiliser une ou plusieurs instances d'un inspecteur multiton, que vous pouvez configurer selon vos besoins. Un inspecteur multiton permet de configurer les paramètres en fonction de conditions spécifiques, comme le réseau, le port et le VLAN. Un ensemble de paramètres pris en charge forme une instance. Un multiton fournit une instance par défaut, et vous pouvez définir des instances supplémentaires basées sur des conditions spécifiques. Si le trafic correspond aux conditions d'une instance personnalisée, les paramètres de cette instance sont appliqués. Sinon, les paramètres de l'instance par défaut sont appliqués. Le nom de l'instance par défaut est le même que le nom de l'inspecteur.

Pour un inspecteur multiton, lorsque vous chargez la configuration remplacée, vous devez également définir une configuration de `binder` correspondante pour chaque instance dans le fichier JSON, sinon le chargement échoue. Vous pouvez également créer de nouvelles instances, mais assurez-vous d'inclure une condition `binder` pour chaque nouvelle instance que vous créez afin d'éviter les erreurs.

Par exemple :

- Inspecteur multiton où l'instance par défaut est modifiée :

```
{
   "http_inspect":{
      "instances":[
         {
            "name":"http_inspect",
            "data":{
               "response_depth":5000
            }
         }
      ]
   }
}
```

- Inspecteur multiton où l'instance par défaut et le `binder` par défaut sont modifiés :

```
{
   "http_inspect":{
      "instances":[
         {
            "name":"http_inspect",
            "data":{
               "response_depth":5000
            }
         }
      ]
   },
   "binder":{
      "rules":[
         {
            "use":{
```

```
                "type":"http_inspect"
            },
            "when":{
                "role":"any",
                "ports":"8080",
                "proto":"tcp",
                "service":"http"
            }
        }
    ]
  }
}
```

- Inspecteur multiton où une instance et un `binder` personnalisés sont ajoutés :

```
{
   "http_inspect":{
      "instances":[
         {
            "name":"http_inspect1",
            "data":{
               "response_depth":5000
            }
         }
      ]
   },
   "binder":{
      "rules":[
         {
            "use":{
               "type":"http_inspect",
               "name":"http_inspect1"
            },
            "when":{
               "role":"any",
               "ports":"8080",
               "proto":"tcp",
               "service":"http"
            }
         }
      ]
   }
}
```

Identification des protocoles et des services dans Snort 3

L'inspecteur `binder` remplit une fonction unique qui a une incidence sur tous les inspecteurs de service Snort. En collaboration avec le module `wizard` (assistant) de Snort, le `binder` détermine quel inspecteur de flux ou de service peut inspecter le trafic réseau. Les configurations contenues dans l'inspecteur `binder` englobent les ports, les hôtes, les CIDR et les services qui déterminent quand un autre inspecteur associé à la même politique d'analyse de réseau doit inspecter le trafic.

Le `wizard` (assistant) prend en charge la configuration des services indépendamment des ports, ce qui permet de détecter les canaux de commande et de contrôle de programmes malveillants.

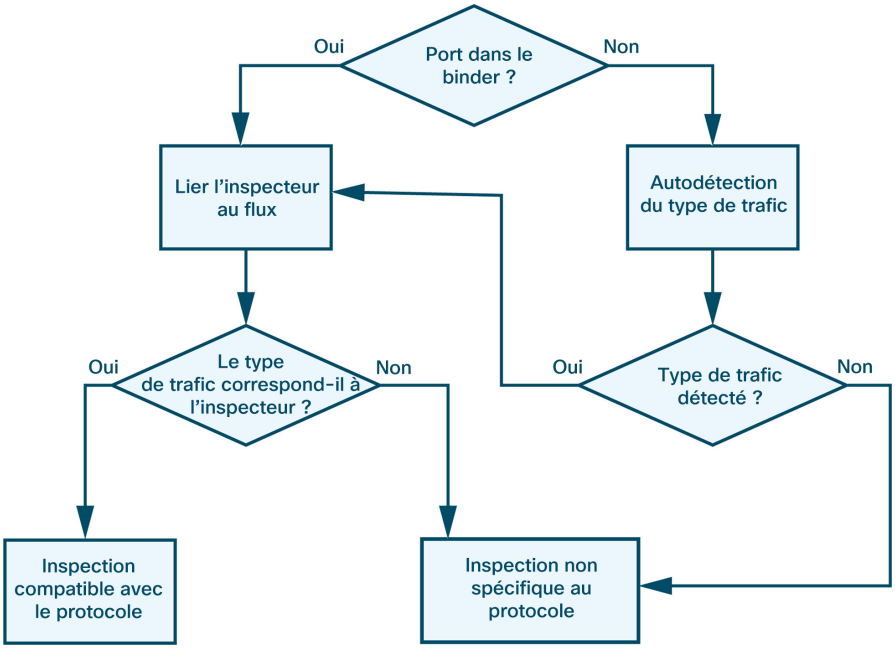
Remarque Vous ne pouvez pas configurer le `wizard` (assistant) à l'aide de l'interface Web Cisco Secure Firewall Management Center.

À l'arrivée du trafic sur un pare-feu, l'inspecteur `binder` recherche les politiques de prévention des intrusions et sélectionne la politique d'accès au réseau (NAP) à appliquer. Dans le cadre d'une NAP, le `binder` identifie les inspecteurs de flux et de service à utiliser pour le flux de données. Par la suite, si le service associé à un flux change, la NAP utilise le `binder` pour sélectionner un autre inspecteur de service.

La configuration de l'inspecteur `binder` comprend des paramètres `when` qui décrivent les caractéristiques du trafic et des paramètres `use` qui spécifient l'inspecteur à appliquer au trafic correspondant à ces caractéristiques. Pour déterminer quel inspecteur appliquer à un flux de données, l'inspecteur `binder` compare le trafic à ses clauses `use` dans l'ordre, de haut en bas, et applique la clause `use` correspondant à la première clause `when` qui correspond au trafic.

Si aucun critère `binder` spécifique ne correspond à un flux de données, le `wizard` (assistant) analyse le flux de données pour déterminer le service. Le `wizard` (assistant) invoque le `binder` pour sélectionner l'inspecteur qui convient à ce service. Si aucun service ne peut être identifié, le `binder` associe généralement un inspecteur de flux au flux, et le système procède à un réassemblage non spécifique au protocole des paquets de données sans tenir compte du contenu de la charge utile.

Le diagramme suivant illustre comment les inspecteurs effectuent une inspection spécifique ou non spécifique au protocole. L'inspection du service dépend de la façon dont vous configurez les paramètres `port`, `host`, `service` et `CIDR` dans l'inspecteur `binder` :



Vous pouvez personnaliser les critères de sélection de l'inspecteur en définissant les paramètres `use` et `when` dans l'inspecteur `binder` d'une NAP par le biais de l'interface Web du centre de gestion. Pour plus de renseignements sur les paramètres `binder`, consultez Présentation de l'inspecteur de binder, à la page 15. Pour plus de renseignements sur la navigation dans l'interface Web du centre de gestion afin de configurer les inspecteurs, consultez le Guide de configuration Cisco Secure Firewall Management Center pour Snort 3.

Une configuration incorrecte du `binder` empêche celui-ci de détecter le service associé au flux ou d'y associer un inspecteur. Si le moteur de règles et la détection automatique ne parviennent pas à comprendre et à identifier le trafic, la configuration d'un critère `when` tel que le port dans l'inspecteur `binder` ne force pas l'inspection. Par exemple, si vous configurez le port 88 du `binder` en tant que port HTTP, le `binder` associe l'inspecteur `http_inspect` à tout flux transitant par ce port. En revanche, si le flux n'est pas de type HTTP, le moteur de règles ne l'inspecte pas en tant que tel, mais procède à une détection basée sur le port.

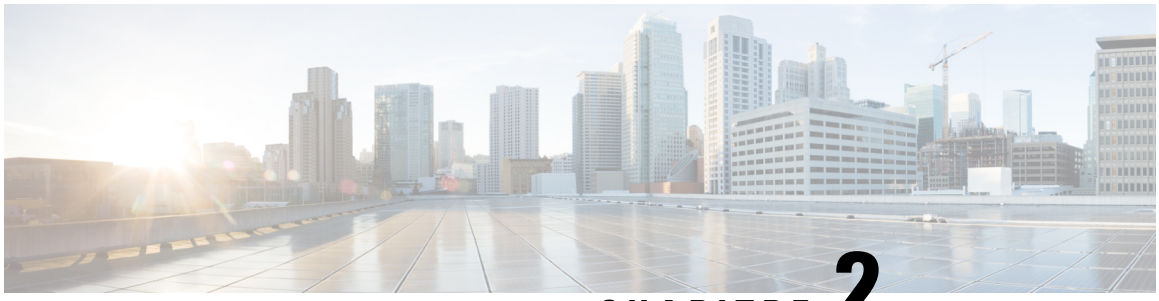
Détection automatique et inspecteurs activés ou désactivés dans la politique d'analyse de réseau

Le comportement de la détection automatique change selon que l'inspecteur ciblé est activé ou désactivé dans la politique d'analyse de réseau. Si l'inspecteur ciblé est activé dans la politique d'analyse de réseau, la détection automatique se déroule comme décrit ci-dessus.

Si l'inspecteur ciblé est désactivé dans la politique d'analyse de réseau, la détection automatique associe généralement au flux un inspecteur de flux (flux TCP ou flux UDP, par exemple). En revanche, le moteur de règles ne procède à aucune inspection ou détection de service. Pour un flux TCP, l'inspecteur de flux TCP procède au réassemblage.

PARTIE I

Inspecteurs Snort 3



CHAPITRE 2

Inspecteur d'usurpation ARP

Présentation de l'inspecteur d'usurpation ARP

Type	Inspecteur (réseau)
Usage	Inspecter
Type d'instance	Singleton
Autres inspecteurs requis	Aucun
Activé	`vrai`

Le protocole ARP (Address Resolution Protocol) est un protocole de communication sans état qui est utilisé au sein d'un réseau unique pour la résolution d'adresses. Lors de l'échange de demandes et de réponses, le protocole ARP n'assure pas l'authentification entre les hôtes.

L'usurpation ARP est un type d'attaque de l'homme du milieu (HDM) qui utilise le protocole ARP au sein d'un réseau local (LAN). Un agresseur perturbe la communication avec un hôte en interceptant les messages destinés à son adresse MAC.

L'inspecteur `arp_spoof` analyse les paquets ARP et détecte les demandes ARP monodiffusion. Pour détecter les attaques par usurpation ARP, l'inspecteur d'usurpation ARP identifie les mappages Ethernet-IP incohérents.

Si cette option est activée, l'inspecteur `arp_spoof` effectue les opérations suivantes :

- Il inspecte les adresses Ethernet et les adresses contenues dans les paquets ARP. En cas d'incohérence, l'inspecteur utilise la règle 112:2 ou 112:3 pour générer des alertes et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.
- Il recherche les demandes ARP monodiffusion. Si une demande ARP monodiffusion est détectée, l'inspecteur utilise la règle 112:1 pour générer des alertes et, dans le cadre d'un déploiement en ligne, supprimer les paquets incriminés.

- Si le paramètre `hosts[]` est spécifié, l'inspecteur utilise ces informations pour détecter les attaques par usurpation ARP. Si une attaque de ce type est détectée, l'inspecteur utilise la règle 112:4 pour générer des alertes et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Paramètres de l'inspecteur d'usurpation ARP

L'inspecteur `arp_spoof` ne fournit pas de valeurs de paramètres de configuration par défaut dans l'interface Web Cisco Secure Firewall Management Center.

Règles de l'inspecteur d'usurpation ARP

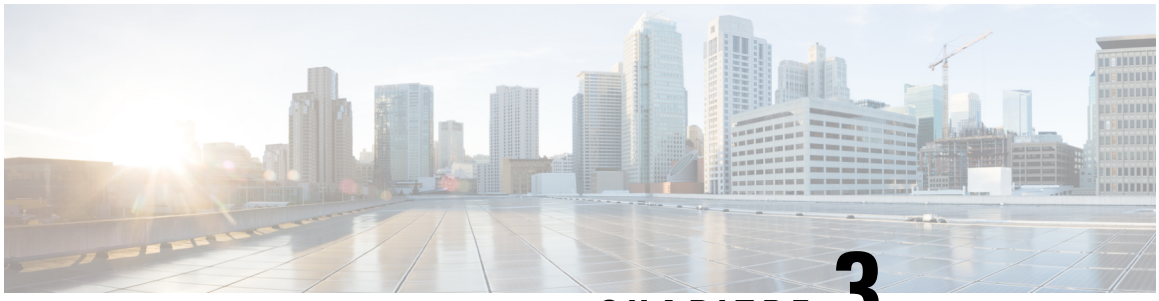
Activez les règles de l'inspecteur `arp_spoof` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 1 : Règles de l'inspecteur d'usurpation ARP

GID:SID	Message de règle
112:1	Demande ARP monodiffusion
112:2	Demande de non-concordance Ethernet/ARP pour la source
112:3	Demande de non-concordance Ethernet/ARP pour la destination
112:4	Tentative d'attaque par usurpation ARP

Options des règles de prévention des intrusions de l'inspecteur d'usurpation ARP

L'inspecteur `arp_spoof` ne comporte aucune option pour les règles de prévention des intrusions.



CHAPITRE 3

Inspecteur Binder

Présentation de l'inspecteur de binder

Type	Inspecteur (passif)
Usage	Inspecter
Type d'instance	Singleton
Autres inspecteurs requis	Selon les liaisons établies
Activé	`vrai`

Un inspecteur `binder` est associé à chaque politique d'analyse de réseau (NAP). Le `binder` détermine quand utiliser un inspecteur de service donné pour inspecter le trafic. Les configurations contenues dans l'inspecteur `binder` englobent les ports, les hôtes, les CIDR et les services qui déterminent quand un autre inspecteur associé à la même NAP doit inspecter le trafic. Lorsqu'une règle `binder` établit une correspondance avec un nouveau flux, l'inspecteur ciblé est associé au flux.

L'inspecteur `binder` peut collaborer avec le `wizard` (assistant) de détection automatique pour configurer les services indépendamment des ports et détecter les canaux de commande et de contrôle de programmes malveillants. Pour en savoir plus, consultez Identification des protocoles et des services dans Snort 3, à la page 7.

Les liaisons sont évaluées au début d'une session, puis à nouveau si un service approprié est identifié dans la session. Les liaisons sont une liste de règles when-use évaluées de haut en bas. Snort utilise les premières configurations de réseau et de service qui correspondent pour inspecter le trafic.

Exemple

Par exemple, si vous souhaitez configurer une NAP pour inspecter le trafic CIP :

- Dans l'inspecteur `binder` de la NAP, mettez à jour la section `"type":"cip"` avec les ports, le rôle et le protocole du trafic à inspecter.
- Passez en revue les valeurs par défaut dans l'inspecteur `cip` de cette même NAP et apportez les modifications nécessaires pour inspecter le trafic CIP.

Voici un exemple de configuration et de liaison `cip`. Cet exemple utilise les options décrites dans Paramètres de l'inspecteur de binder, à la page 18.

```
{
   "use": {
     "type":"cip"
   },
   "when": {
     "proto":"udp",
     "ports":"22222 33333",
     "role":"server"
   }
 },
 {
   "use": {
     "type":"cip"
   },
   "when": {
     "role":"server",
     "ports":"44818",
     "proto":"tcp"
   }
 },
```

Détection automatique des services pour une configuration sans port

Le `wizard` (assistant) de détection automatique permet de configurer les services indépendamment des ports, et de détecter les canaux de commande et de contrôle de programmes malveillants. À l'arrivée du trafic, l'inspecteur `binder` associe le `wizard` (assistant) de détection automatique au flux et vérifie la charge utile initiale pour identifier le service utilisé par le trafic. Par exemple, `GET` indique HTTP et `Hello` indique SMTP. Une fois le service identifié, Snort associe l'inspecteur de service approprié au flux et dissocie le `wizard` (assistant) de détection automatique de celui-ci.

Remarque Vous ne pouvez pas configurer le `wizard` (assistant) de détection automatique à l'aide de l'interface Web Cisco Secure Firewall Management Center.

Si le moteur de règles et le `wizard` (assistant) de détection automatique ne parviennent pas à comprendre et à identifier le trafic, la configuration d'un port dans l'inspecteur `binder` ne force pas l'inspection.

Configuration de la détection automatique et du « binder »

L'inspecteur `binder` compare les règles de prévention des intrusions dans l'ordre, de haut en bas, et applique la première règle correspondant au trafic. Si vous n'avez pas configuré l'inspecteur `binder` pour le service détecté dans le flux, l'assistant de détection automatique peut toujours associer le flux à l'inspecteur approprié. Par exemple :

- Si la charge utile est `GET` et que l'assistant de détection automatique identifie le type de trafic comme étant HTTP, l'inspecteur `binder` associe l'inspecteur HTTP à ce flux.
- Si le type de trafic ne peut pas être identifié, le moteur de règles effectue une inspection non spécifique au protocole.

Une configuration de port incorrecte empêche l'inspecteur `binder` de détecter automatiquement le service associé à ce flux et d'associer un inspecteur à celui-ci. Par exemple, si vous configurez le port 88 du « binder » en tant que port HTTP, l'inspecteur `binder` associe l'inspecteur HTTP à tout flux transitant par ce port. En revanche, si le flux n'est pas de type HTTP, le moteur de règles ne l'inspecte pas en tant que tel. En conséquence, l'inspection et la détection expirent.

Détection automatique et activation ou désactivation des inspecteurs dans la politique d'analyse de réseau

Le comportement de la détection automatique change selon que l'inspecteur ciblé est activé ou désactivé dans la politique d'analyse de réseau. Si l'inspecteur ciblé est activé dans la politique d'analyse de réseau, la détection automatique se déroule normalement.

Si l'inspecteur ciblé est désactivé dans la politique d'analyse de réseau, la détection automatique associe généralement au flux un inspecteur de flux (flux TCP ou flux UDP, par exemple). En revanche, le moteur de règles ne procède à aucune inspection ou détection de service. Pour un flux TCP, l'inspecteur de flux TCP procède au réassemblage.

Bonnes pratiques en matière de configuration de l'inspecteur de classeur

Tenez compte des bonnes pratiques suivantes lors de la configuration de l'inspecteur binder :

- Ne configurez pas les ports dans l'inspecteur binder, sauf si cela est nécessaire pour cet inspecteur. La configuration des ports n'améliore pas l'efficacité si le moteur de règles détecte automatiquement le trafic. Toutefois, une configuration incorrecte des ports peut empêcher la détection des évasions.
- Configurez un port pour un seul inspecteur. Si un port est configuré deux fois dans le binder pour des protocoles et des inspecteurs différents, il déclenchera automatiquement le premier inspecteur.
- Ajoutez la configuration d'un inspecteur de service à l'inspecteur `binder` si vous ne la voyez pas dans la configuration par défaut de l'inspecteur `binder`. Par exemple, si vous souhaitez utiliser l'inspecteur `cip`, ajoutez les options `use` et `when` de l'inspecteur `cip` dans le binder.
- Pour l'inspecteur de flux TCP, configurez les réseaux de manière à établir une liaison personnalisée avec les configurations du système d'exploitation. Ces configurations réseau s'appliquent à tous les ports.
- Pour les inspecteurs de service, évitez les liaisons rigides de ports si le binder détecte automatiquement le protocole dans le flux. Si le protocole n'est pas détectable, une liaison rigide ne garantit pas la détection et l'inspection.

Inspecteurs nécessitant une configuration des ports

Configurez les ports dans l'inspecteur binder pour les inspecteurs suivants, car les protocoles concernés ne permettent pas la détection automatique :

- `cip`

- gtp_inspect
- iec104
- modbus
- s7commplus

Inspecteurs ne nécessitant pas de configuration de ports

Ne configurez pas de ports dans l'inspecteur binder pour les inspecteurs suivants, car les protocoles concernés permettent la détection automatique :

- arp_spoof
- dce_smb
- dce_tcp
- dnp3
- ftp_client
- ftp_server
- http_inspect
- imap
- normalizer
- pop
- port_scan
- sip
- smtp
- ssh
- stream_icmp
- stream_ip
- stream_tcp
- stream_udp
- telnet

Paramètres de l'inspecteur de binder

binder[]

Un « binder » comprend un ensemble de règles définies sous forme de paires d'objets `when` et `use`.

Type : tableau

Exemple :

```
{
    binder: {
        rules: [
            {
                "when": {
                    ...
                },
                "use": {
                    ...
                 }
            },
            {
                "when": {
                    ...
                },
                "use": {
                    ...
                }
            }
        ]
    }
}
```

binder[].use.type

Spécifie l'inspecteur à associer au flux de données lorsque les critères du paramètre `when` sont satisfaits. Par exemple, pour inspecter le trafic CIP, ajoutez `use.type` avec la valeur `cip`.

Type : chaîne

Valeurs valides : nom d'un inspecteur Snort 3 décrit dans ce document.

Valeur par défaut : l'inspecteur `binder` comprend un paramètre `use.type` pour chaque inspecteur pris en charge.

binder[].when.proto

Spécifie le protocole que le trafic doit utiliser pour associer le flux de données à l'inspecteur désigné dans `use.type`. Par exemple, si la politique d'analyse de réseau est configurée pour inspecter le trafic TCP, ce paramètre doit être défini sur `tcp` dans l'inspecteur `binder`.

Type : énumération

Valeurs valides : `any`, `ip`, `icmp`, `tcp`, `udp`, `user`, `file`

Valeur par défaut : l'inspecteur `binder` comprend un paramètre `when.proto` pour chaque protocole.

binder[].when.ports

Spécifie les ports que le trafic doit utiliser pour associer le flux de données à l'inspecteur désigné dans `use.type`. Par exemple, pour inspecter le trafic sur le port TCP 80, définissez `when.proto` sur `tcp` et `when.ports` sur `80`.

Établissez une liste d'un ou de plusieurs ports représentés par des entiers décimaux ou hexadécimaux. Utilisez un espace comme séparateur entre les différents ports et entourez la liste de guillemets doubles.

Type : chaîne

Plage valide : de `1` à `65 535`

Valeur par défaut : `65 535` (cette valeur peut varier selon la valeur de `when.proto`.)

binder[].when.role

Spécifie les rôles que le trafic doit utiliser pour associer le flux à l'inspecteur désigné dans `use.type`.

Type : énumération

Valeurs valides : `client`, `server`, `any`

Valeur par défaut : `any`

Spécifie le service que le trafic doit utiliser pour associer le flux à l'inspecteur désigné dans `use.type`.

Type : chaîne

Valeurs valides : nom d'un service qui peut encapsuler des données entrantes, par exemple : `netbios-ssn` ou `dcerpc`.

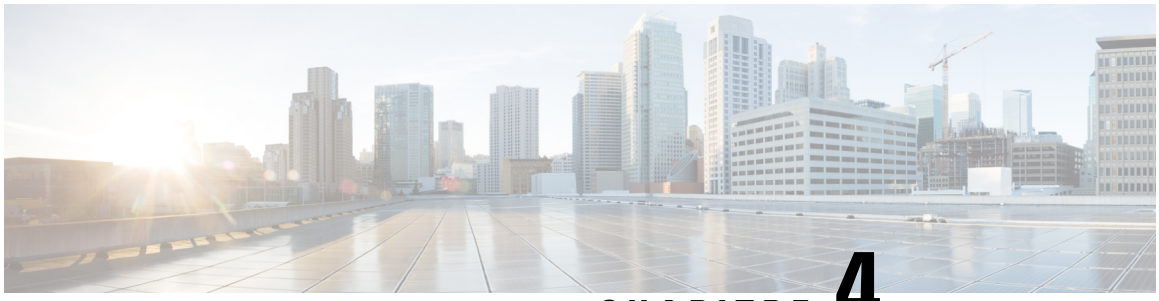
Valeur par défaut : aucune

Règles de l'inspecteur de classeur

Aucune règle n'est associée à l'inspecteur `binder`.

Options des règles de prévention des intrusions de l'inspecteur de binder

L'inspecteur `binder` ne comporte aucune option pour les règles de prévention des intrusions.



CHAPITRE 4

Inspecteur CIP

Présentation de l'inspecteur CIP

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`stream_tcp`
Activé	`faux`

Le protocole CIP (Common Industrial Protocol) est un protocole d'application qui prend en charge les applications d'automatisation industrielle. EtherNet/IP (ENIP) est une implémentation de CIP utilisée sur les réseaux Ethernet.

L'inspecteur `cip` détecte le trafic CIP et ENIP s'exécutant sur TCP ou UDP et l'envoie au moteur de règles de prévention des intrusions. Vous pouvez utiliser les mots-clés CIP et ENIP dans les règles de prévention des intrusions personnalisées pour détecter les attaques dans le trafic CIP et ENIP.

Remarque Dans Snort 3, l'inspecteur `cip` ne prend pas en charge les détecteurs d'applications CIP. Pour implémenter la détection d'applications CIP, vous pouvez créer et importer des règles de prévention des intrusions CIP personnalisées et activer les règles IPS appropriées. Pour plus de renseignements, consultez la documentation de configuration de Snort 3 relative à votre application de gestion.

Bonnes pratiques en matière de configuration de l'inspecteur CIP

Tenez compte des bonnes pratiques suivantes lors de la configuration de l'inspecteur `cip` :

- Vous devez ajouter le port de détection CIP par défaut (44818) et tous les autres ports CIP nécessaires dans l'inspecteur `binder`.
- Nous vous recommandons d'utiliser une action de prévention des intrusions comme action par défaut de votre stratégie de contrôle d'accès.
- Pour détecter les applications CIP et ENIP, vous devez activer l'inspecteur `cip` dans la politique d'analyse de réseau personnalisée correspondante.
- Pour bloquer le trafic des applications CIP ou ENIP à l'aide de règles de contrôle d'accès, assurez-vous que l'inspecteur de normalisation et son option de mode en ligne sont activés (paramètre par défaut) dans la politique d'analyse de réseau correspondante.
- Pour abandonner le trafic qui déclenche les règles de l'inspecteur `cip` et les règles de prévention des intrusions CIP, assurez-vous que l'option **Drop when Inline** (Abandonner quand en ligne) est activée dans la politique de prévention des intrusions correspondante.
- L'inspecteur `cip` ne prend pas en charge les actions par défaut suivantes d'une stratégie de contrôle d'accès :
  - **Contrôle d'accès : faire confiance à tout le trafic**
  - **Contrôle d'accès : bloquer tout le trafic**
- L'inspecteur `cip` ne prend pas en charge la visibilité des applications CIP, y compris la découverte de réseau.

Paramètres de l'inspecteur CIP

Configuration du port TCP CIP

L'inspecteur `binder` configure le port TCP CIP. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
    {
        "when": {
                    "role": "server",
                    "proto": "tcp",
                    "ports": "44818"
        },
        "use": {
                    "type": "cip"
        }
    }
]
```

embedded_cip_path

Détermine si l'inspecteur vérifie le chemin de connexion CIP intégré.

Type : chaîne

Valeurs valides :

- `"false"`
- Chemin CIP entre guillemets doubles, par exemple, `"0x2 0x36"`.

Valeur par défaut : `"false"`

unconnected_timeout

Définit le délai d'expiration par défaut, en secondes, pour les connexions non établies. Lorsqu'un message de demande CIP ne contient pas de valeur d'expiration spécifique au protocole et que le Nombre maximal de demandes non connectées simultanées par connexion TCP est atteint, le système temporise le message pendant le nombre de secondes spécifié par cette option. À l'expiration de la temporisation, le message est supprimé pour libérer de l'espace pour les demandes futures.

Lorsque vous spécifiez 0, tout le trafic pour lequel aucun délai spécifique au protocole n'a été configuré expire en premier.

Type : entier

Plage valide : de `0` à `360`

Valeur par défaut : `300`

max_unconnected_messages

Définit le nombre maximal de messages CIP non connectés simultanés par connexion TCP. Si le système atteint le nombre maximal de demandes simultanées qui peuvent rester sans réponse, il met fin à la connexion.

Type : entier

Plage valide : de `1` à `10 000`

Valeur par défaut : `100`

max_cip_connections

Définit le nombre maximal de connexions CIP simultanées autorisées par le système, par connexion TCP.

Type : entier

Plage valide : de `1` à `10 000`

Valeur par défaut : `100`

Règles de l'inspecteur CIP

Activez les règles de l'inspecteur `cip` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 2 : Règles de l'inspecteur CIP

GID:SID	Message de règle
148:1	Données CIP mal formées
148:2	Données CIP non conformes à la norme ODVA
148:3	Limite de connexions CIP dépassée. La connexion la moins récemment utilisée a été supprimée
148:4	Limite de demandes CIP non connectées dépassée. La demande la plus ancienne a été supprimée

Options des règles de prévention des intrusions de l'inspecteur CIP

cip_attribute

Paramètre de détection pour la recherche de correspondance avec l'attribut CIP.

Type : intervalle

Syntaxe : `cip_attribute: <range_operator><positive integer>;` ou `cip_attribute: <positive integer><range_operator><positive integer>;`

Valeurs valides : un ensemble d'un ou plusieurs entiers compris entre 0 et 65 535, et un opérateur `range_operator`, comme spécifié dans le Tableau 3 : Formats de plages.

Exemples : `cip_attribute: <100;`

cip_class

Paramètre de détection pour la recherche de correspondance avec la classe CIP.

Type : intervalle

Syntaxe : `cip_class: <range_operator><positive integer>;` ou `cip_class: <positive integer><range_operator><positive integer>;`

Valeurs valides : un ensemble d'un ou plusieurs entiers compris entre 0 et 65 535, et un opérateur `range_operator`, comme spécifié dans le Tableau 3 : Formats de plages.

Exemples : `cip_class: <25;`

cip_conn_path_class

Paramètre de détection pour la recherche de correspondance avec la classe de chemin de connexion CIP.

Type : intervalle

Syntaxe : `cip_conn_path_class: <range_operator><positive integer>;` ou `cip_conn_path_class: <positive integer><range_operator><positive integer>;`

Valeurs valides : un ensemble d'un ou plusieurs entiers compris entre 0 et 65 535, et un opérateur `range_operator`, comme spécifié dans le Tableau 3 : Formats de plages.

Exemples : `cip_conn_path_class: <85;`

cip_instance

Paramètre de détection pour la recherche de correspondance avec l'instance CIP.

Type : intervalle

Syntaxe : `cip_instance: <range_operator><positive integer>;` ou `cip_instance: <positive integer><range_operator><positive integer>;`

Valeurs valides : un ensemble d'un ou plusieurs entiers compris entre 0 et 65 535, et un opérateur `range_operator`, comme spécifié dans le Tableau 3 : Formats de plages.

Exemples : `cip_instance: <15;`

cip_req

Paramètre de détection pour la recherche de correspondance avec la demande CIP.

Syntaxe : `cip_req;`

Exemples : `cip_req;`

cip_rsp

Paramètre de détection pour la recherche de correspondance avec la réponse CIP.

Syntaxe : `cip_rsp;`

Exemples : `cip_rsp;`

service_cip

Paramètre de détection pour la recherche de correspondance avec le service CIP.

Type : intervalle

Syntaxe : `cip_service: <range_operator><positive integer>;` ou `cip_service: <positive integer><range_operator><positive integer>;`

Valeurs valides : un ensemble d'un ou plusieurs entiers compris entre 0 et 127, et un opérateur `range_operator`, comme spécifié dans le Tableau 3 : Formats de plages.

Exemples : `cip_service: <50;`

cip_status

Paramètre de détection pour la recherche de correspondance avec l'état CIP.

Type : intervalle

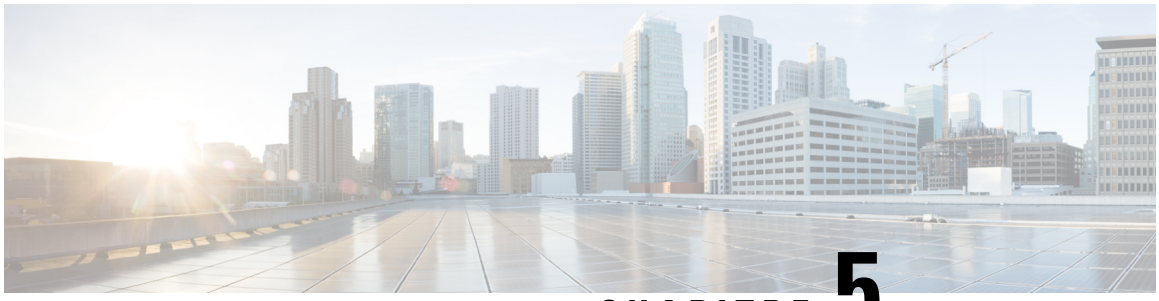
Syntaxe : `cip_status: <range_operator><positive integer>;` ou `cip_status: <positive integer><range_operator><positive integer>;`

Valeurs valides : un ensemble d'un ou plusieurs entiers compris entre 0 et 255, et un opérateur `range_operator`, comme spécifié dans le Tableau 3 : Formats de plages.

Exemples : `cip_status: <250;`

Tableau 3 : Formats de plages

Format de plage	Opérateur	Description
opérateur i		
	<	Supérieur à
	>	Supérieur à
	=	Égal à
	≠	Différent de
	≤	Inférieur ou égal à
	≥	Supérieur ou égal à
j opérateur k		
	<>	Supérieur à j et inférieur à k
	<=>	Supérieur ou égal à j et inférieur ou égal à k



CHAPITRE 5

Inspecteur SMB DCE

Présentation de l'inspecteur SMB DCE

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	Aucun
Activé	`vrai`

Le protocole DCE/RPC permet aux processus se trouvant sur des hôtes réseau distincts de communiquer comme si les processus se trouvaient sur le même hôte. Ces communications interprocessus sont généralement transportées entre les hôtes sur TCP et UDP. Dans le cadre du transport TCP, DCE/RPC peut également être encapsulé dans le protocole SMB (Server Message Block) de Windows ou dans Samba, une implémentation ouverte (Open Source) de SMB utilisée pour la communication entre processus dans les environnements mixtes comprenant des systèmes d'exploitation Windows, UNIX ou Linux.

Bien que la plupart des exploits DCE/RPC se produisent dans les demandes des clients DCE/RPC ciblant les serveurs DCE/RPC, qui peuvent être pratiquement n'importe quel hôte de votre réseau qui exécute Windows ou Samba, des exploits peuvent également se produire dans les réponses des serveurs.

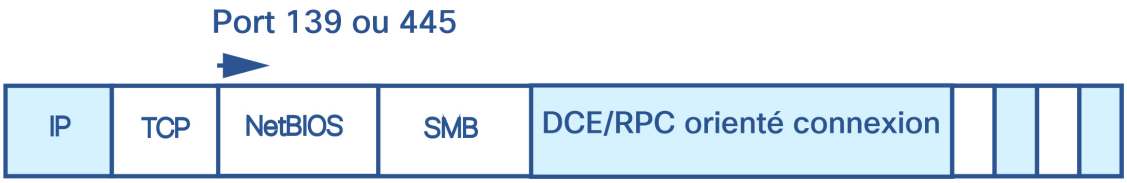
IP encapsule tous les transports DCE/RPC. TCP transporte tous les DCE/RPC orientés connexion, tels que SMB.

L'inspecteur `dce_smb` détecte les DCE/RPC orientés connexion dans le protocole SMB et utilise des caractéristiques spécifiques au protocole, comme la longueur de l'en-tête et l'ordre des fragments de données, pour :

- détecter les demandes et les réponses DCE/RPC encapsulées dans les transferts SMB,

- analyser les flux de données DCE/RPC, et détecter les comportements anormaux et les techniques d'évasion dans le trafic DCE/RPC,
- analyser les flux de données SMB, et détecter les comportements anormaux et les techniques d'évasion,
- désegmenter SMB et défragmenter DCE/RPC,
- normaliser le trafic DCE/RPC pour le traitement par le moteur de règles.

Le diagramme suivant illustre le moment où l'inspecteur SMB DCE commence à traiter le trafic pour le transport SMB.



L'inspecteur `dce_smb` reçoit généralement le trafic SMB sur le port TCP 139 (service de session NetBIOS) ou sur le port Windows 445 (qui implémente un service similaire). Étant donné que SMB remplit de nombreuses fonctions autres que le transport de DCE/RPC, l'inspecteur teste d'abord si le trafic SMB transporte du trafic DCE/RPC, et arrête le traitement si ce n'est pas le cas ou le poursuit dans le cas inverse.

Les descriptions des paramètres et des fonctionnalités de l'inspecteur `dce_smb` comprennent l'implémentation Microsoft de DCE/RPC, connue sous le nom de MSRPC (Microsoft Remote Procedure Call; appel de procédure distante Microsoft), ainsi que SMB et Samba.

Politiques basées sur la cible

Les implémentations de Windows et Samba DCE/RPC sont très différentes. Par exemple, toutes les versions de Windows utilisent l'ID de contexte DCE/RPC dans le premier fragment lors de la défragmentation du trafic DCE/RPC, et toutes les versions de Samba utilisent l'ID de contexte dans le dernier fragment. Autre exemple, Windows Vista utilise le champ d'en-tête `opnum` (numéro d'opération) dans le premier fragment pour identifier un appel de fonction spécifique, tandis que Samba et toutes les autres versions de Windows utilisent le champ `opnum` dans le dernier fragment.

Il existe des différences importantes entre les implémentations SMB de Windows et de Samba. Par exemple, Windows reconnaît les commandes SMB OPEN et READ lorsqu'il utilise des canaux nommés, mais Samba ne reconnaît pas ces commandes.

C'est pourquoi l'inspecteur `dce_smb` utilise une approche basée sur la cible. Lorsque vous configurez une instance de l'inspecteur `dce_smb`, le paramètre `policy` spécifie une implémentation particulière du protocole SMB DCE/RPC. Ce paramètre, combiné aux informations relatives à l'hôte, établit une politique de serveur basée sur la cible par défaut. Vous pouvez également configurer des inspecteurs supplémentaires qui ciblent d'autres hôtes et les implémentations SMB DCE/RPC. L'implémentation SMB DCE/RPC spécifiée par la politique de serveur basée sur la cible par défaut s'applique à tout hôte non ciblé par une autre instance de l'inspecteur `dce_smb`.

Les implémentations SMB DCE/RPC que l'inspecteur `dce_smb` peut cibler avec le paramètre `policy` sont les suivantes :

- WinXP (par défaut)
- Win2000
- WinVista

- Win2003
- Win2008
- Win7
- Samba
- Samba-3.0.37
- Samba-3.0.22
- Samba-3.0.20

Inspection des fichiers

L'inspecteur `dce_smb` prend en charge l'inspection des fichiers pour les versions 1, 2 et 3 de SMB.

L'inspecteur `dce_smb` examine les transferts de fichiers SMB normaux. Cela implique la vérification du type de fichier et de sa signature lors du traitement de celui-ci, ainsi que la définition d'un pointeur pour l'option de règle `file_data`. L'inspecteur `dce_smb` prend en charge l'inspection des transferts de fichiers SMB normaux pour les versions 1, 2 et 3 de SMB lorsqu'il est utilisé en coordination avec l'inspecteur `file_id` (décrit dans la documentation ouverte (Open Source) de Snort 3, disponible à l'adresse https://www.snort.org/snort3). Pour activer l'inspection des fichiers, configurez l'inspecteur `file_id` selon les besoins, et définissez les paramètres `dce_smb` `smb_file_inspection` et `smb_file_ depth`. Le paramètre `smb_file_ depth` indique le nombre d'octets de données de fichier que l'inspecteur `file_id` examinera à partir du pointeur spécifié par l'option de règle IPS `file_data`. Pour plus de renseignements, consultez la documentation ouverte (Open Source) de Snort 3, disponible à l'adresse https://www.snort.org/snort3.

Défragmentation

L'inspecteur `dce_smb` prend en charge le réassemblage des paquets de données fragmentés. Cette fonctionnalité est particulièrement utile en mode en ligne pour détecter les exploits dès le début du processus d'inspection, avant que la défragmentation complète ne soit effectuée, ou pour détecter les exploits qui tirent parti de la fragmentation pour dissimuler leur présence. Sachez que la désactivation de la défragmentation peut entraîner un grand nombre de faux négatifs.

Paramètres de l'inspecteur SMB DCE

Configuration du port SMB DCE

L'inspecteur `binder` configure le port SMB DCE. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
   {
       "when": {
           "role":"any",
           "service":"netbios-ssn",
           "ports": ""
       },
        "use": {
            "type":"dce_smb"
```

```
        }
    }
]
```

max_frag_len

Spécifie la longueur maximale des fragments (en octets) pour la défragmentation. Pour traiter les fragments de grande taille, l'inspecteur analyse le contenu du paquet jusqu'à la taille spécifiée avant de le défragmenter.

Remarque La valeur spécifiée dans ce paramètre doit être supérieure ou égale à la profondeur d'analyse nécessaire aux règles pour assurer la détection. Pour que toutes les données soient soumises à la détection, utilisez la valeur par défaut.

Type : entier

Plage valide : de `1 514` à `65 535`

Valeur par défaut : `65 535`

smb_max_compound

Spécifie le nombre maximal de commandes à traiter dans une demande SMB.

Type : entier

Plage valide : de `0` à `255`

Valeur par défaut : `3`

smb_max_chain

Spécifie le nombre maximal de commandes SMB AndX en chaîne autorisé. En règle générale, plusieurs commandes AndX en chaîne représentent un comportement anormal et peuvent indiquer une tentative d'évitement. Spécifiez `1` pour n'autoriser aucune commande en chaîne ou `0` pour désactiver la détection du nombre de commandes en chaîne.

L'inspecteur `dce_smb` commence par compter le nombre de commandes en chaîne et génère un événement si les règles associées de l'inspecteur SMB sont activées et que le nombre de commandes en chaîne est égal ou supérieur à la valeur configurée. Le traitement se poursuit ensuite.

Vous pouvez activer la règle 133:20 afin de générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés en lien avec ce paramètre.

Type : entier

Plage valide : de `0` à `255`

Valeur par défaut : `3`

disable_defrag

Spécifie s'il faut défragmenter le trafic DCE/RPC fragmenté. Lorsqu'il est activé, l'inspecteur `dce_smb` détecte les anomalies et envoie les données DCE/RPC au moteur de règles, mais au risque de manquer des exploits dans les données DCE/RPC fragmentées.

Bien que le paramètre `disable_defrag` permette de ne pas défragmenter le trafic et d'accélérer le traitement, il convient de noter que la plupart des exploits DCE/RPC tentent de tirer parti de la fragmentation pour

dissimuler leur présence. L'activation de ce paramètre contournerait la plupart des exploits connus, ce qui entraînerait un grand nombre de faux négatifs.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

limit_alerts

Indique si les alertes DCE doivent être limitées à un maximum d'une par signature et par flux.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `true`

reassemble_threshold

Spécifie le nombre minimal d'octets qui doivent être mis en file d'attente dans les tampons de désegmentation et de défragmentation DCE/RPC avant d'envoyer un paquet réassemblé au moteur de règles. Ce paramètre est particulièrement utile en mode en ligne, car il peut permettre de détecter un exploit à un stade précoce, avant qu'une défragmentation complète ne soit effectuée.

Notez qu'une valeur faible augmente la probabilité d'une détection précoce, mais peut avoir un impact négatif sur les performances. Si vous activez ce paramètre, vous devez tester son incidence sur les performances.

La valeur `0` désactive le réassemblage.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `0`

politique

Spécifie l'implémentation Windows ou Samba DCE/RPC utilisée par les hôtes ciblés sur votre segment de réseau supervisé.

Type : énumération

Valeurs valides : une chaîne sélectionnée dans la liste suivante : `Win2000`, `WinXP`, `WinVista`, `Win2003`, `Win2008`, `Win7`, `Samba`, `Samba-3.0.37`, `Samba-3.0.22`, `Samba-3.0.20`

Valeur par défaut : `WinXP`

smb_max_credit

Spécifie le nombre maximal de demandes en attente.

Type : entier

Plage valide : de `1` à `65 536`

Valeur par défaut : `8 192`

smb_file_depth

Spécifie le nombre d'octets inspectés lorsqu'un fichier est détecté dans le trafic SMB, en commençant à l'emplacement spécifié par l'option de règle IPS `file_data` (décrite dans la documentation ouverte (Open Source) de Snort 3, disponible à l'adresse https://www.snort.org/snort3).

Spécifiez `-1` pour désactiver l'inspection des fichiers.

Spécifiez `0` pour une inspection de fichiers illimitée.

Type : entier

Plage valide : de `-1` à `32 767`

Valeur par défaut : `16 384`

Lorsqu'un fichier est détecté dans le trafic SMB, le paramètre `smb_file_ depth` indique le nombre d'octets de données de fichier que l'inspecteur examinera à partir du pointeur défini dans l'option de règle IPS `file_data`. Pour inspecter le type de fichier et la signature, `dce_smb` utilise les paramètres `enable_type`, `type_depth`, `enable_signature` et `signature_depth` définis dans l'inspecteur `file_id`. Pour plus de renseignements sur l'inspecteur `file_id`, consultez la documentation ouverte (Open Source) de Snort, disponible à l'adresse https://www.snort.org/snort3.

memcap

Spécifie la limite maximale de mémoire (en octets) allouée à l'inspecteur. Lorsque le plafond de mémoire est atteint ou dépassé, l'inspecteur `dce_smb` supprime les données les moins récemment utilisées pour libérer de l'espace.

Type : entier

Plage valide : de `512` à `9 007 199 254 740 992 (maxSZ)`

Valeur par défaut : `8 388 608`

smb_fingerprint_policy

L'inspecteur détecte la version de Windows ou de Samba identifiée dans les demandes et réponses de `Session Setup AndX`. Lorsque la version détectée est différente de la version de Windows ou de Samba configurée pour le paramètre `policy` de l'inspecteur, la version détectée remplace la version configurée, mais seulement pour la durée de la session en cours.

Par exemple, si vous définissez `policy` sur Windows XP et que l'inspecteur détecte Windows Vista, il utilise une politique Windows Vista pendant cette session. Les autres paramètres restent en vigueur.

Type : énumération

Valeurs valides : `none`, `client`, `server` ou `both`

- Utilisez `client` afin d'inspecter le trafic serveur-client pour le type de politique.
- Utilisez `server` afin d'inspecter le trafic client-serveur pour le type de politique.
- Utilisez `both` afin d'inspecter le trafic serveur-client et client-serveur pour le type de politique.
- Utilisez `none` pour désactiver l'inspection de la version de Windows ou de Samba.

Valeur par défaut : `none`

smb_legacy_mode

Lorsque `smb_legacy_mode` est défini sur `true`**,** le système n'applique les règles de prévention des intrusions SMB qu'au trafic SMB version 1, tandis que les règles DCE/RPC ciblent le trafic DCE/RPC utilisant SMB version 1 comme protocole de transport.

Lorsque `smb_legacy_mode` est défini sur `false`, le système applique les règles de prévention des intrusions SMB au trafic utilisant les versions 1 et 2 de SMB, et :

- Concernant les versions 7.0 et 7.0.x, le système n'applique les règles de prévention des intrusions DCE/RPC au trafic DCE/RPC utilisant SMB comme protocole de transport que pour la version 1 de SMB.
- Concernant les versions 7.1 et ultérieures, le système applique les règles de prévention des intrusions DCE/RPC au trafic DCE/RPC utilisant SMB comme protocole de transport pour les versions 1 et 2 de SMB.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

valid_smb_versions

Spécifie les versions de SMB à inspecter. Utilisez un espace comme séparateur entre les différentes versions de SMB.

Type : chaîne

Valeurs valides : `v1`, `v2`, `v3`, `all`

Valeur par défaut : `all`

Règles de l'inspecteur SMB DCE

Activez les règles de l'inspecteur `dce_smb` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 4 : Règles de l'inspecteur SMB DCE

GID:SID	Message de règle
133:2	SMB – Type de session de service NetBIOS incorrect
133:3	SMB – Type de message SMB incorrect
133:4	SMB – ID SMB incorrect (autre que \xffSMB pour SMB1 ou que \xfeSMB pour SMB2)
133:5	SMB – Nombre de mots incorrect ou taille de structure incorrecte
133:6	SMB – Nombre d'octets incorrect
133:7	SMB – Type de format incorrect

GID:SID	Message de règle
133:8	SMB – Décalage incorrect
133:9	SMB – Nombre total de données nul
133:10	SMB – Longueur des données NetBIOS inférieure à la longueur de l'en-tête SMB
133:11	SMB – Longueur restante des données NetBIOS inférieure à la longueur de la commande
133:12	SMB – Longueur restante des données NetBIOS inférieure au nombre d'octets de la commande
133:13	SMB – Longueur restante des données NetBIOS inférieure à la taille des données de la commande
133:14	SMB – Nombre total de données restantes inférieur à la taille des données de cette commande
133:15	SMB – Nombre total de données envoyées (STDu64) supérieur au nombre total de données attendu par la commande
133:16	SMB – Nombre d'octets inférieur à la taille des données de la commande (STDu64)
133:17	SMB – Taille des données de la commande non valide pour le nombre d'octets
133:18	SMB – Nombre excessif de demandes de connexion à l'arborescence avec des réponses de connexion à l'arborescence en attente
133:19	SMB – Nombre excessif de demandes de lecture avec des réponses de lecture en attente
133:20	SMB – Chaînage de commandes excessif
133:21	SMB – Demandes multiples de connexion en chaîne
133:22	SMB – Demandes multiples de connexion à l'arborescence en chaîne
133:23	SMB – Connexion en chaîne/composée suivie d'une déconnexion
133:24	SMB – Connexion à l'arborescence en chaîne/composée, suivie d'une déconnexion de l'arborescence
133:25	SMB – Ouverture de canal en chaîne/composée suivie d'une fermeture de canal
133:26	SMB – Accès au partage non valide
133:44	SMB – SMB version 1 non valide détectée
133:45	SMB – SMB version 2 non valide détectée
133:46	SMB – Utilisateur, connexion à l'arborescence, liaison de fichier non valide
133:47	SMB – Composition excessive de commandes

GID:SID	Message de règle
133:48	SMB – Nombre de données nul
133:50	SMB – Nombre maximal de demandes en attente dépassé
133:51	SMB – Demandes en attente avec le même MID
133:52	SMB – Dialecte obsolète négocié
133:53	SMB – Commande obsolète utilisée
133:54	SMB – Commande inhabituelle utilisée
133:55	SMB – Nombre de configurations non valide pour la commande
133:56	SMB – Le client a tenté plusieurs négociations de dialecte sur la session
133:57	SMB – Le client a tenté de créer ou de définir les attributs d'un fichier en lecture seule/masqué/système
133:58	SMB – Le décalage de fichier fourni est supérieur à la taille de fichier spécifiée
133:59	SMB – La commande suivante spécifiée dans l'en-tête SMB2 dépasse la limite de la charge utile

Options des règles de prévention des intrusions de l'inspecteur DCE

dce_iface

Spécifie les éléments suivants, séparés par des virgules :

- L'UUID d'une interface de service.
- La version de l'interface (facultatif). Le paramètre par défaut correspond à n'importe quelle version.
- Un indicateur permettant de déterminer si une règle doit trouver une correspondance sur n'importe quel fragment d'une demande (facultatif). Par défaut, la règle ne s'applique qu'au premier fragment.

Dans le cadre du protocole DCE/RPC, un client doit établir une liaison avec un service avant de pouvoir l'appeler. Lorsqu'un client envoie une demande de liaison au serveur, il peut spécifier une ou plusieurs interfaces de service. Chaque interface est représentée par un UUID, et chaque UUID d'interface est associé à un index unique (ou ID de contexte) que les demandes futures peuvent utiliser pour référencer le service appelé par le client. Le serveur répond en indiquant les UUID d'interface qu'il considère comme valides et permet au client d'adresser des demandes à ces services. Lorsqu'un client effectue une demande, il spécifie l'ID de contexte afin que le serveur sache à quel service la demande est adressée.

L'option de règle `dce_iface` permet à une règle de demander à l'inspecteur si le client a établi une liaison avec un UUID d'interface donné et si la demande du client est destinée à cette interface. Cela peut éliminer les faux-positifs lorsque plusieurs services sont associés avec succès, car l'inspecteur peut corréler l'UUID de liaison à l'ID de contexte utilisé dans la demande.

L'option `dce_iface` requiert que l'inspecteur assure le suivi des demandes Bind et Alter Context du client ainsi que des réponses Bind Ack et Alter Context du serveur dans le cadre du DCE/RPC orienté connexion. Pour chaque demande Bind et Alter Context, le client spécifie une liste d'UUID d'interface ainsi qu'un identifiant (ou ID de contexte) pour chaque UUID d'interface qui sera utilisé pendant la session DCE/RPC pour référencer l'interface. La réponse du serveur indique les interfaces auxquelles il autorise le client à adresser des demandes. Il approuve ou refuse la liaison du client à une interface donnée. Ce suivi permet de mettre en corrélation l'ID de contexte d'une demande avec l'UUID de l'interface qu'il représente lors du traitement.

L'option de règle `dce_iface` trouve une correspondance si :

- l'UUID de l'interface spécifiée correspond à l'UUID de l'interface (telle qu'elle est désignée par l'ID de contexte) de la demande DCE/RPC,

et

- l'argument `version` n'est pas fourni, ou l'argument `version` est fourni et correspond à l'UUID de l'interface de la demande DCE/RPC,

et

- l'argument `any_frag` est fourni, ou l'argument `any_frag` est absent et l'option `dce_iface` correspond aux critères d'UUID et de version du fragment de demande initial.

Exemples :

```
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188;
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188, <2;
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188,any_frag;
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188, =1, any_frag;
```

dce_iface.uuid

Dans une demande DCE/RPC, il est possible de spécifier si les UUID sont représentés en gros boutiste ou en petit boutiste. La représentation de l'UUID de l'interface dans une demande est différente selon le boutisme spécifié. L'inspecteur `dce_rpc` normalise l'UUID. Cela signifie que la spécification de l'UUID dans l'option de règle `dce_iface` doit être écrite de la même manière, quel que soit le boutisme de la demande.

Par exemple, un UUID serait représenté comme suit dans une demande de liaison petit boutiste :

```
|f8 91 7b 5a 00 ff d0 11 a9 b2 00 c0 4f b6 e6 fc|
```

et ce même UUID serait représenté comme suit dans une demande de liaison gros boutiste :

```
|5a 7b 91 f8 ff 00 11 d0 a9 b2 00 c0 4f b6 e6 fc|
```

Dans les règles Snort 3 qui emploient l'option `dce_iface`, l'UUID doit être représenté en gros-boutiste sous forme de chaîne, quel que soit le boutisme de la demande :

```
5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc
```

Type : chaîne

Syntaxe : `dce_iface: <UUID>;`

Valeurs valides : où l'*`UUID`* est composé de 32 chiffres hexadécimaux, affichés en cinq groupes séparés par des traits d'union, sous la forme `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`

Exemples : `dce_iface: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc;`

dce_iface.version

Une version est associée à chaque interface de service. Certaines versions peuvent ne pas être vulnérables à certains exploits. Par conséquent, vous pouvez spécifier un ou plusieurs numéros de version dans l'option `dce_iface` pour déterminer s'il est nécessaire de rechercher la présence d'un exploit particulier.

Type : intervalle

Syntaxe : `dce_iface: <range_operator><positive integer>;` ou `dce_iface: <positive integer><range_operator><positive integer>;`

Valeurs valides : un ensemble d'un ou plusieurs numéros de version positifs et un opérateur `range_operator`, comme spécifié dans le Tableau 5 : Formats de plages.

Exemples : `dce_iface: =6;`

dce_iface.any_frag

Une demande DCE/RPC peut être divisée en un ou plusieurs fragments. Des indicateurs sont définis dans l'en-tête DCE/RPC pour déterminer si le fragment actuel correspond au premier fragment, à un fragment intermédiaire ou au dernier fragment de la demande. La mise en œuvre de nombreuses vérifications de données au sein de la demande DCE/RPC n'est pertinente que si la demande DCE/RPC porte sur un premier fragment (ou une demande complète). Ainsi, les fragments qui suivent le premier fragment contiennent des données situées plus loin dans la demande DCE/RPC. Par exemple, une règle conçue pour examiner les cinq premiers octets de la demande (comme un champ de longueur) ne fonctionnera pas correctement si elle est appliquée à un fragment autre que le premier, car les données recherchées ne s'y trouveront pas. Le début des fragments suivants est décalé d'une certaine longueur par rapport au début de la demande. Cela peut être une source de faux-positifs dans le trafic DCE/RPC fragmenté.

De ce fait, l'inspecteur DCE/RPC est configuré par défaut pour ne cibler que le fragment initial d'une demande. Pour forcer l'inspecteur à examiner tous les fragments d'une demande afin de trouver une correspondance, ajoutez `any_frag` à l'option de règle `dce_iface`. Notez qu'une demande DCE/RPC défragmentée est considérée comme une demande complète.

Syntaxe : `dec_iface: any_frag;`

Exemples : `dce_iface: any_frag;`

dce_opnum

Établit une correspondance avec un numéro d'opération d'appel RPC DCE, une plage de numéros d'opération ou une liste de numéros d'opération. Cette option représente un ou plusieurs appels de fonction spécifiques qui peuvent être adressés à une interface. Une fois qu'un client a établi une liaison avec une interface de service donnée et lui a envoyé une demande, des règles doivent déterminer l'appel de fonction que le client adresse au service pour y rechercher la présence d'éventuels exploits. Les appels de fonctions sont spécifiés sous la forme d'une liste de numéros d'opération (opnums) entre guillemets doubles.

Type : chaîne

Syntaxe : `dce_opnum: <opnum_list>;`

Où *`opnum_list`* correspond à l'un des éléments suivants :

- Un seul entier
- Une liste d'entiers séparés par des virgules
- Une plage d'entiers spécifiés avec un trait d'union séparant le nombre le plus petit et le nombre le plus grand de cette plage

- Une combinaison des éléments ci-dessus

Valeurs valides : une liste d'opnums de demande DCE/RPC.

Exemples :

```
dce_opnum: "15";
dce_opnum: "15-18";
dce_opnum: "15, 18-20";
dce_opnum: "15, 17, 20-22";
```

dce_stub_data

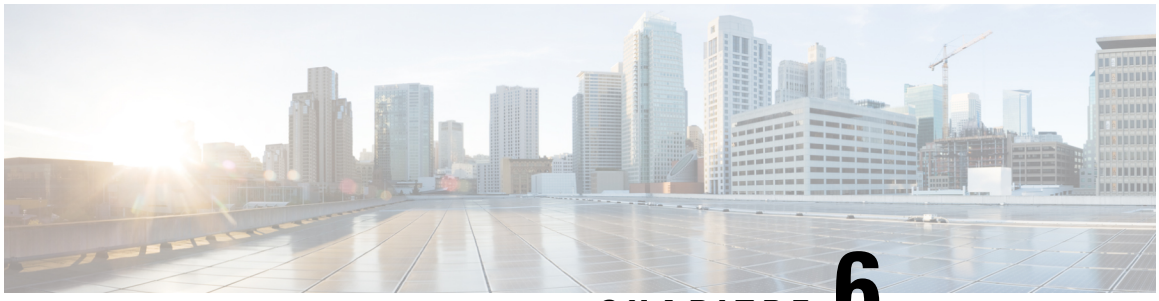
Cette option place le curseur de détection (utilisé pour parcourir la charge utile des paquets lors du traitement des règles) au début des données de stub DCE/RPC, quelles que soient les options de règles précédentes. Cette option s'applique si des données de stub DCE/RPC sont présentes.

Syntaxe : `dce_stub_data;`

Exemples : `dce_stub_data;`

Tableau 5 : Formats de plages

Format de plage	Opérateur	Description
opérateur i		
	<	Supérieur à
	>	Supérieur à
	=	Égal à
	≠	Différent de
	≤	Inférieur ou égal à
	≥	Supérieur ou égal à
j opérateur k		
	<>	Supérieur à j et inférieur à k
	<=>	Supérieur ou égal à j et inférieur ou égal à k



CHAPITRE 6

Inspecteur TCP DCE

Présentation de l'inspecteur TCP DCE

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	Aucun
Activé	`vrai`

Le protocole DCE/RPC permet aux processus se trouvant sur des hôtes réseau distincts de communiquer comme si les processus se trouvaient sur le même hôte. Ces communications interprocessus sont généralement transportées entre les hôtes sur TCP. Dans le cadre du transport TCP, DCE/RPC peut également être encapsulé dans le protocole SMB (Server Message Block) de Windows ou dans Samba, une implémentation ouverte (Open Source) de SMB utilisée pour la communication entre processus dans les environnements mixtes comprenant des systèmes d'exploitation Windows, UNIX ou Linux.

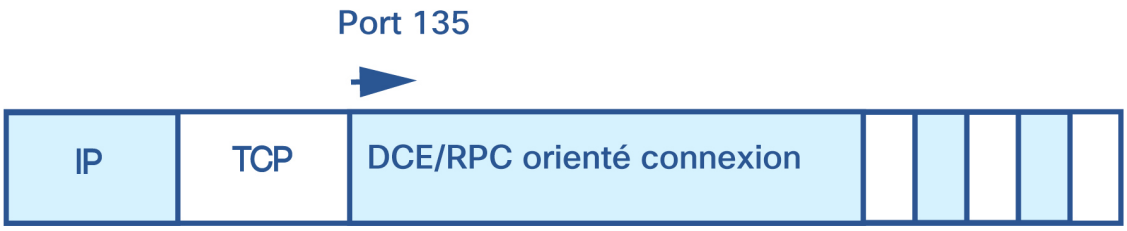
Bien que la plupart des exploits DCE/RPC se produisent dans les demandes des clients DCE/RPC ciblant les serveurs DCE/RPC, qui peuvent être pratiquement n'importe quel hôte de votre réseau qui exécute Windows ou Samba, des exploits peuvent également se produire dans les réponses des serveurs.

IP encapsule tous les transports DCE/RPC. TCP transporte tous les DCE/RPC en mode connexion. L'inspecteur TCP DCE détecte les DCE/RPC orientés connexion et utilise des caractéristiques spécifiques au protocole (comme la longueur de l'en-tête et l'ordre des fragments de données) pour :

- détecter les demandes et les réponses DCE/RPC encapsulées dans des transports TCP, y compris celles qui utilisent RPC sur HTTP version 1,
- analyser les flux de données DCE/RPC, et détecter les comportements anormaux et les techniques d'évasion dans le trafic DCE/RPC,

- défragmenter le DCE/RPC,
- normaliser le trafic DCE/RPC pour le traitement par le moteur de règles.

Le diagramme suivant illustre le moment où l'inspecteur TCP DCE commence à traiter le trafic pour le transport TCP.



Le port TCP 135, bien connu, identifie le trafic DCE/RPC dans le transport TCP. La figure n'inclut pas l'appel RPC sur HTTP. Pour les appels RPC sur HTTP, le protocole ETCD/RPC orienté connexion est transporté directement sur TCP, comme le montre la figure, après une séquence de configuration initiale sur HTTP.

Politiques basées sur la cible

Les implémentations de Windows et Samba DCE/RPC sont très différentes. Par exemple, toutes les versions de Windows utilisent l'ID de contexte DCE/RPC dans le premier fragment lors de la défragmentation du trafic DCE/RPC, et toutes les versions de Samba utilisent l'ID de contexte dans le dernier fragment. À titre d'autre exemple, Windows XP utilise le champ d'en-tête « opnum » (numéro d'opération) dans le premier fragment pour identifier un appel de fonction spécifique, et Samba et toutes les autres versions de Windows utilisent le champ « opnum » dans le dernier fragment.

C'est pourquoi l'inspecteur `dce_tcp` utilise une approche basée sur la cible. Lorsque vous configurez une instance de l'inspecteur `dce_tcp`, le paramètre `policy` spécifie une implémentation particulière du protocole TCP DCE/RPC. Ce paramètre, combiné aux informations relatives à l'hôte, établit une politique de serveur basée sur la cible par défaut. Vous pouvez également configurer des inspecteurs supplémentaires qui ciblent d'autres hôtes et les implémentations TCP DCE/RPC. L'implémentation TCP DCE/RPC spécifiée par la politique de serveur basée sur la cible par défaut s'applique à tout hôte non ciblé par une autre instance de l'inspecteur `dce_tcp`.

Les implémentations DCE/RPC que l'inspecteur TCP DCE peut cibler avec le paramètre `policy` sont les suivantes :

- WinXP (par défaut)
- Win2000
- WinVista
- Win2003
- Win2008
- Win7
- Samba
- Samba-3.0.37
- Samba-3.0.22
- Samba-3.0.20

Défragmentation

L'inspecteur TCP DCE prend en charge le réassemblage des paquets de données fragmentés avant de les envoyer au moteur de détection. Cette fonctionnalité est particulièrement utile en mode en ligne pour détecter les exploits dès le début du processus d'inspection, avant que la défragmentation complète ne soit effectuée, ou pour détecter les exploits qui tirent parti de la fragmentation pour dissimuler leur présence. Sachez que la désactivation de la défragmentation peut entraîner un grand nombre de faux négatifs.

Paramètres de l'inspecteur TCP DCE

Configuration du port TCP DCE

L'inspecteur `binder` configure le port TCP DCE. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
   {
        "when": {
            "role": "any",
            "proto": "tcp",
            "service": "dcerpc",
            "ports": ""
        },
        "use": {
            "type": "dce_tcp"
        }
    }
]
```

max_frag_len

Spécifie la longueur maximale des fragments (en octets) pour la défragmentation. Pour traiter les fragments de grande taille, l'inspecteur analyse le contenu du paquet jusqu'à la taille spécifiée avant de le défragmenter.

Remarque La valeur spécifiée dans ce paramètre doit être supérieure ou égale à la profondeur d'analyse nécessaire aux règles pour assurer la détection. Pour que toutes les données soient soumises à la détection, utilisez la valeur par défaut.

Type : entier

Plage valide : de `1 514` à `65 535`

Valeur par défaut : `65 535`

disable_defrag

Spécifie s'il faut défragmenter le trafic DCE/RPC fragmenté. Lorsque ce paramètre est défini sur `true`, l'inspecteur détecte toujours les anomalies et envoie les données DCE/RPC au moteur de règles, mais avec un risque accru de manquer les exploits présents dans les données DCE/RPC fragmentées.

Bien que ce paramètre permette de ne pas défragmenter le trafic et d'accélérer le traitement, il convient de noter que la plupart des exploits DCE/RPC tentent de tirer parti de la fragmentation pour dissimuler leur

présence. L'activation de ce paramètre contournerait la plupart des exploits connus, ce qui entraînerait un grand nombre de faux négatifs.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

limit_alerts

Indique si les alertes DCE doivent être limitées à un maximum d'une par signature et par flux.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `true`

reassemble_threshold

Spécifie le nombre minimal d'octets qui doivent être mis en file d'attente dans les tampons de désegmentation et de défragmentation DCE/RPC avant d'envoyer un paquet réassemblé au moteur de règles. Ce paramètre est particulièrement utile en mode en ligne, car il peut permettre de détecter un exploit à un stade précoce, avant qu'une défragmentation complète ne soit effectuée.

Une valeur faible augmente la probabilité d'une détection précoce, mais peut avoir un impact négatif sur les performances. Si vous activez ce paramètre, vous devez tester son incidence sur les performances.

La valeur `0` désactive le réassemblage.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `0`

politique

Spécifie l'implémentation Windows ou Samba DCE/RPC utilisée par les hôtes ciblés sur votre segment de réseau supervisé.

Type : énumération

Valeurs valides : une chaîne sélectionnée dans la liste suivante : `Win2000`, `WinXP`, `WinVista`, `Win2003`, `Win2008`, `Win7`, `Samba`, `Samba-3.0.37`, `Samba-3.0.22`, `Samba-3.0.20`

Valeur par défaut : `WinXP`

Règles de l'inspecteur TCP DCE

Activez les règles de l'inspecteur `dce_tcp` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 6 : Règles de l'inspecteur TCP DCE

GID:SID	Message de règle
133:27	DCE/RPC orienté connexion – Version majeure non valide
133:28	DCE/RPC orienté connexion – Version mineure non valide
133:29	DCE/RPC orienté connexion – Type de PDU non valide
133:30	DCE/RPC orienté connexion – Longueur du fragment inférieure à la taille de l'en-tête
133:31	DCE/RPC orienté connexion – Longueur du fragment restant inférieure à la taille nécessaire
133:32	DCE/RPC orienté connexion – Aucun élément de contexte spécifié
133:33	DCE/RPC orienté connexion – Aucune syntaxe de transfert spécifiée
133:34	DCE/RPC orienté connexion – Longueur sur un fragment non final inférieure à la taille maximale de transmission de fragment négociée pour le client
133:35	DCE/RPC orienté connexion – Longueur du fragment supérieure à la taille maximale de transmission de fragment négociée
133:36	DCE/RPC orienté connexion – Ordre des octets du contexte de modification différent de celui de la liaison
133:37	DCE/RPC orienté connexion – ID d'appel d'un fragment non premier ou dernier différent de celui établi pour la demande fragmentée
133:38	DCE/RPC orienté connexion – Opnum d'un fragment non premier ou dernier différent de celui établi pour la demande fragmentée
133:39	DCE/RPC orienté connexion – ID de contexte d'un fragment non premier ou dernier différent de celui établi pour la demande fragmentée

Options des règles de prévention des intrusions de l'inspecteur DCE

dce_iface

Spécifie les éléments suivants, séparés par des virgules :

- L'UUID d'une interface de service.
- La version de l'interface (facultatif). Le paramètre par défaut correspond à n'importe quelle version.
- Un indicateur permettant de déterminer si une règle doit trouver une correspondance sur n'importe quel fragment d'une demande (facultatif). Par défaut, la règle ne s'applique qu'au premier fragment.

Dans le cadre du protocole DCE/RPC, un client doit établir une liaison avec un service avant de pouvoir l'appeler. Lorsqu'un client envoie une demande de liaison au serveur, il peut spécifier une ou plusieurs interfaces de service. Chaque interface est représentée par un UUID, et chaque UUID d'interface est associé à un index unique (ou ID de contexte) que les demandes futures peuvent utiliser pour référencer le service appelé par le client. Le serveur répond en indiquant les UUID d'interface qu'il considère comme valides et permet au client d'adresser des demandes à ces services. Lorsqu'un client effectue une demande, il spécifie l'ID de contexte afin que le serveur sache à quel service la demande est adressée.

L'option de règle `dce_iface` permet à une règle de demander à l'inspecteur si le client a établi une liaison avec un UUID d'interface donné et si la demande du client est destinée à cette interface. Cela peut éliminer les faux-positifs lorsque plusieurs services sont associés avec succès, car l'inspecteur peut corréler l'UUID de liaison à l'ID de contexte utilisé dans la demande.

L'option `dce_iface` requiert que l'inspecteur assure le suivi des demandes Bind et Alter Context du client ainsi que des réponses Bind Ack et Alter Context du serveur dans le cadre du DCE/RPC orienté connexion. Pour chaque demande Bind et Alter Context, le client spécifie une liste d'UUID d'interface ainsi qu'un identifiant (ou ID de contexte) pour chaque UUID d'interface qui sera utilisé pendant la session DCE/RPC pour référencer l'interface. La réponse du serveur indique les interfaces auxquelles il autorise le client à adresser des demandes. Il approuve ou refuse la liaison du client à une interface donnée. Ce suivi permet de mettre en corrélation l'ID de contexte d'une demande avec l'UUID de l'interface qu'il représente lors du traitement.

L'option de règle `dce_iface` trouve une correspondance si :

- l'UUID de l'interface spécifiée correspond à l'UUID de l'interface (telle qu'elle est désignée par l'ID de contexte) de la demande DCE/RPC,

et

- l'argument `version` n'est pas fourni, ou l'argument `version` est fourni et correspond à l'UUID de l'interface de la demande DCE/RPC,

et

- l'argument `any_frag` est fourni, ou l'argument `any_frag` est absent et l'option `dce_iface` correspond aux critères d'UUID et de version du fragment de demande initial.

Exemples :

```
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188;
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188, <2;
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188,any_frag;
dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188, =1, any_frag;
```

dce_iface.uuid

Dans une demande DCE/RPC, il est possible de spécifier si les UUID sont représentés en gros boutiste ou en petit boutiste. La représentation de l'UUID de l'interface dans une demande est différente selon le boutisme spécifié. L'inspecteur `dce_rpc` normalise l'UUID. Cela signifie que la spécification de l'UUID dans l'option de règle `dce_iface` doit être écrite de la même manière, quel que soit le boutisme de la demande.

Par exemple, un UUID serait représenté comme suit dans une demande de liaison petit boutiste :

```
|f8 91 7b 5a 00 ff d0 11 a9 b2 00 c0 4f b6 e6 fc|
```

et ce même UUID serait représenté comme suit dans une demande de liaison gros boutiste :

```
|5a 7b 91 f8 ff 00 11 d0 a9 b2 00 c0 4f b6 e6 fc|
```

Dans les règles Snort 3 qui emploient l'option `dce_iface`, l'UUID doit être représenté en gros-boutiste sous forme de chaîne, quel que soit le boutisme de la demande :

```
5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc
```

Type : chaîne

Syntaxe : `dce_iface: <UUID>;`

Valeurs valides : où l'*`UUID`* est composé de 32 chiffres hexadécimaux, affichés en cinq groupes séparés par des traits d'union, sous la forme `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`

Exemples : `dce_iface: 5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc;`

dce_iface.version

Une version est associée à chaque interface de service. Certaines versions peuvent ne pas être vulnérables à certains exploits. Par conséquent, vous pouvez spécifier un ou plusieurs numéros de version dans l'option `dce_iface` pour déterminer s'il est nécessaire de rechercher la présence d'un exploit particulier.

Type : intervalle

Syntaxe : `dce_iface: <range_operator><positive integer>;` ou `dce_iface: <positive integer><range_operator><positive integer>;`

Valeurs valides : un ensemble d'un ou plusieurs numéros de version positifs et un opérateur `range_operator`, comme spécifié dans le Tableau 7 : Formats de plages.

Exemples : `dce_iface: =6;`

dce_iface.any_frag

Une demande DCE/RPC peut être divisée en un ou plusieurs fragments. Des indicateurs sont définis dans l'en-tête DCE/RPC pour déterminer si le fragment actuel correspond au premier fragment, à un fragment intermédiaire ou au dernier fragment de la demande. La mise en œuvre de nombreuses vérifications de données au sein de la demande DCE/RPC n'est pertinente que si la demande DCE/RPC porte sur un premier fragment (ou une demande complète). Ainsi, les fragments qui suivent le premier fragment contiennent des données situées plus loin dans la demande DCE/RPC. Par exemple, une règle conçue pour examiner les cinq premiers octets de la demande (comme un champ de longueur) ne fonctionnera pas correctement si elle est appliquée à un fragment autre que le premier, car les données recherchées ne s'y trouveront pas. Le début des fragments suivants est décalé d'une certaine longueur par rapport au début de la demande. Cela peut être une source de faux-positifs dans le trafic DCE/RPC fragmenté.

De ce fait, l'inspecteur DCE/RPC est configuré par défaut pour ne cibler que le fragment initial d'une demande. Pour forcer l'inspecteur à examiner tous les fragments d'une demande afin de trouver une correspondance, ajoutez `any_frag` à l'option de règle `dce_iface`. Notez qu'une demande DCE/RPC défragmentée est considérée comme une demande complète.

Syntaxe : `dec_iface: any_frag;`

Exemples : `dce_iface: any_frag;`

dce_opnum

Établit une correspondance avec un numéro d'opération d'appel RPC DCE, une plage de numéros d'opération ou une liste de numéros d'opération. Cette option représente un ou plusieurs appels de fonction spécifiques qui peuvent être adressés à une interface. Une fois qu'un client a établi une liaison avec une interface de

service donnée et lui a envoyé une demande, des règles doivent déterminer l'appel de fonction que le client adresse au service pour y rechercher la présence d'éventuels exploits. Les appels de fonctions sont spécifiés sous la forme d'une liste de numéros d'opération (opnums) entre guillemets doubles.

Type : chaîne

Syntaxe : `dce_opnum: <opnum_list>;`

Où *opnum_list* correspond à l'un des éléments suivants :

- Un seul entier
- Une liste d'entiers séparés par des virgules
- Une plage d'entiers spécifiés avec un trait d'union séparant le nombre le plus petit et le nombre le plus grand de cette plage
- Une combinaison des éléments ci-dessus

Valeurs valides : une liste d'opnums de demande DCE/RPC.

Exemples :

```
dce_opnum: "15";
dce_opnum: "15-18";
dce_opnum: "15, 18-20";
dce_opnum: "15, 17, 20-22";
```

dce_stub_data

Cette option place le curseur de détection (utilisé pour parcourir la charge utile des paquets lors du traitement des règles) au début des données de stub DCE/RPC, quelles que soient les options de règles précédentes. Cette option s'applique si des données de stub DCE/RPC sont présentes.

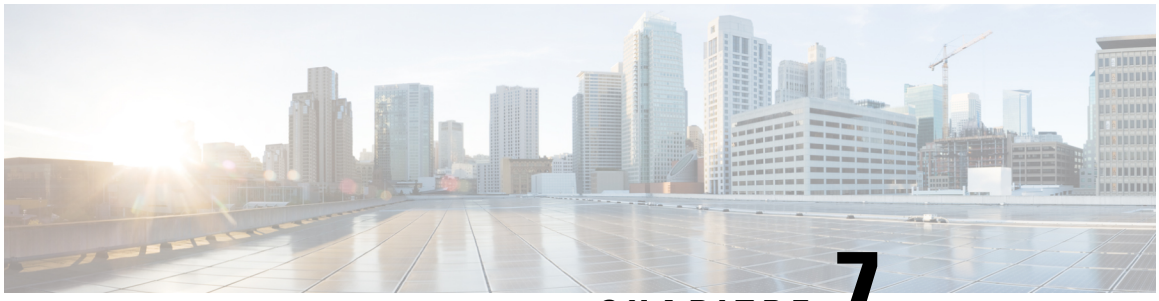
Syntaxe : `dce_stub_data;`

Exemples : `dce_stub_data;`

Tableau 7 : Formats de plages

Format de plage	Opérateur	Description
opérateur i		
	<	Supérieur à
	>	Supérieur à
	=	Égal à
	≠	Différent de
	≤	Inférieur ou égal à
	≥	Supérieur ou égal à
j opérateur k		

Format de plage	Opérateur	Description
	<>	Supérieur à j et inférieur à k
	<=>	Supérieur ou égal à j et inférieur ou égal à k



CHAPITRE 7

Inspecteur DNP3

Présentation de l'inspecteur DNP3

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`stream_tcp`, `stream_udp`
Activé	`faux`

Le protocole DNP3 (Distributed Network Protocol) est un protocole SCADA (Supervisory Control and Data Acquisition) initialement développé pour assurer une communication cohérente entre les postes électriques. DNP3 est couramment utilisé dans les secteurs de l'eau, du traitement des déchets et des transports.

L'inspecteur `dnp3` détecte les anomalies au sein du trafic DNP3 et analyse le protocole DNP3. Les options de règles de prévention des intrusions `dnp3` permettent d'accéder à certains champs du protocole DNP3.

Paramètres de l'inspecteur DNP3

Configuration du port TCP DNP3

L'inspecteur `binder` configure le port TCP DNP3. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
   {
       "when": {
           "role": "any",
```

```
            "service": "dnp3"
    },
       "use": {
            "type": "dnp3"
       }
    }
]
```

check_crc

Indique si les sommes de contrôle contenues dans les trames de couche liaison DNP3 doivent être validées. L'inspecteur `dnp3` ignore les trames contenant des sommes de contrôle non valides. Si la règle de prévention des intrusions 145:1 est activée, Snort génère des alertes en cas de sommes de contrôle non valides.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

Règles de l'inspecteur DNP3

Activez les règles de l'inspecteur `dnp3` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 8 : Règles de l'inspecteur DNP3

GID:SID	Message de règle
145:1	La trame de couche liaison DNP3 contient un CRC incorrect
145:2	La trame de couche liaison DNP3 a été rejetée
145:3	Le segment de la couche transport DNP3 a été abandonné pendant le réassemblage
145:4	Le tampon de réassemblage DNP3 a été effacé sans réassemblage d'un message complet
145:5	La trame de couche liaison DNP3 utilise une adresse réservée
145:6	Le fragment de couche application DNP3 utilise un code de fonction réservé

Options des règles de prévention des intrusions de l'inspecteur DNP3

dnp3_data

Le mot clé `dnp3_data` positionne le curseur de détection au début des données DNP3 dans un fragment de couche application, indépendamment des options de règles précédentes. Cette option vous permet de créer des règles basées sur les données contenues dans les fragments sans avoir à diviser ces données et à ajouter des CRC tous les 16 octets.

Syntaxe : `dnp3_data;`

Exemples : `dnp3_data;`

dnp3_func

Cette option recherche une correspondance avec le code de fonction contenu dans un en-tête de demande/réponse de la couche application DNP3. Le code peut être un nombre décimal ou une chaîne de la liste ci-dessous.

Type : chaîne

Syntaxe : `dnp3_func: <DNP3_function>;`

Valeurs valides : *`DNP3_function`* peut être l'une des valeurs suivantes :

- Un nombre entier compris entre `0` et `255`
- `confirm` (correspond au code de fonction `0`)
- `read` (correspond au code de fonction `1`)
- `write` (correspond au code de fonction `2`)
- `select` (correspond au code de fonction `3`)
- `operate` (correspond au code de fonction `4`)
- `direct_operate` (correspond au code de fonction `5`)
- `direct_operat_nr` (correspond au code de fonction `6`)
- `immed_freeze` (correspond au code de fonction `7`)
- `immed_freeze_nr` (correspond au code de fonction `8`)
- `freeze_clear` (correspond au code de fonction `9`)
- `freeze_clear_nr` (correspond au code de fonction `10`)
- `freeze_at_time` (correspond au code de fonction `11`)
- `freeze_at_time_nr` (correspond au code de fonction `12`)
- `cold_restart` (correspond au code de fonction `13`)
- `warm_restart` (correspond au code de fonction `14`)
- `initialize_data` (correspond au code de fonction `15`)
- `initialize_appl` (correspond au code de fonction `16`)
- `start_appl` (correspond au code de fonction `17`)
- `stop_appl` (correspond au code de fonction `18`)
- `save_config` (correspond au code de fonction `19`)
- `enable_unsolicited` (correspond au code de fonction `20`)
- `disable_unsolicited` (correspond au code de fonction `21`)
- `assign_class` (correspond au code de fonction `22`)

- `delay_measure` (correspond au code de fonction `23`)
- `record_current_time` (correspond au code de fonction `24`)
- `open_file` (correspond au code de fonction `25`)
- `close_file` (correspond au code de fonction `26`)
- `delete_file` (correspond au code de fonction `27`)
- `get_file_info` (correspond au code de fonction `28`)
- `authenticate_file` (correspond au code de fonction `29`)
- `abort_file` (correspond au code de fonction `30`)
- `activate_config` (correspond au code de fonction `31`)
- `authenticate_req` (correspond au code de fonction `32`)
- `authenticate_err` (correspond au code de fonction `33`)
- `response` (correspond au code de fonction `129`)
- `unsolicited_response` (correspond au code de fonction `130`)
- `authenticate_resp` (correspond au code de fonction `131`)

Exemples :

```
dnp3_func: 1;
```

```
dnp3_func: delete_file;
```

dnp3_ind

Fournissez une liste d'indicateurs internes à comparer avec ceux d'un en-tête de réponse de couche application DNP3. Si vous fournissez plusieurs indicateurs au sein d'une même option, la règle se déclenche lorsque l'un des indicateurs est défini. Pour générer une alerte en cas de détection de plusieurs indicateurs, utilisez plusieurs options de règles.

Type : chaîne

Syntaxe : `dnp3_ind: "<flag> <flag>";`

Valeurs valides : un ou plusieurs indicateurs internes DNP3, où l'indicateur `flag` peut être l'une des valeurs suivantes :

- `all_stations`
- `class_1_events`
- `class_2_events`
- `class_3_events`
- `need_time`
- `local_control`
- `device_trouble`

- `device_restart`
- `no_func_code_support`
- `object_unknown`
- `parameter_error`
- `event_buffer_overflow`
- `already_executing`
- `config_corrupt`
- `reserved_2`
- `reserved_1`

Exemples :

Alerte au redémarrage de l'appareil OU au lancement de la synchronisation de l'heure :

```
dnp3_ind:"device_restart need_time";
```

Alerte sur les événements class_1 ET class_2 ET class_3 :

```
dnp3_ind:class_1_events; dnp3_ind:class_2_events; dnp3_ind:class_3_events;
```

dnp3_obj

Recherche des correspondances dans les groupes d'en-têtes d'objets DNP3 et leurs variantes.

Type : entier

Syntaxe: `dnp3_obj:<groupnum>,<varnum>;`

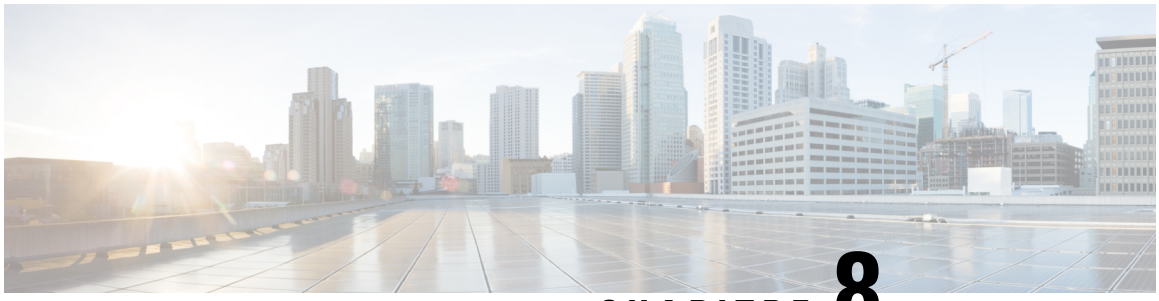
Valeurs valides : identifiants des groupes d'objets DNP3 et identifiants de leurs variantes, où :

- *`groupnum`* est un entier compris entre `0` et `255` qui spécifie un groupe d'objets DNP3,
- *`varnum`* est un entier compris entre `0` et `255` qui spécifie une variante dans le groupe d'objets.

Exemples :

Alerte sur l'objet DNP3 `Date and Time` :

```
dnp3_obj:50,1;
```



CHAPITRE 8

Inspecteur de client FTP

Présentation de l'inspecteur de client FTP

Type	Inspecteur (passif)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`ftp_server`, `stream_tcp`
Activé	`vrai`

Le protocole FTP (File Transfer Protocol) est un protocole réseau utilisé pour transférer des fichiers entre des clients et des serveurs sur un réseau TCP/IP. Dès qu'une connexion est établie entre un client et un serveur, le client envoie des commandes au serveur pour charger ou télécharger des fichiers, et interprète les réponses reçues du serveur.

L'inspecteur `ftp_client` examine et normalise les réponses sur le canal de commande FTP.

À partir d'un tampon de canal de commande FTP, l'inspecteur `ftp_client` interprète les codes de réponse et les messages FTP. L'inspecteur `ftp_client` vérifie la validité des paramètres. En outre, il détermine quand une connexion de commande FTP est chiffrée et quand un canal de données FTP est ouvert.

Paramètres de l'inspecteur de client FTP

bounce

Indique si les rebonds FTP doivent être vérifiés en examinant les informations relatives à l'hôte dans les commandes `ftp port` émises par le client. Lorsque le paramètre `bounce` est défini sur `true`, si les informations d'hôte contenues dans une commande `ftp port` ne correspondent pas à l'adresse IP ou aux informations

d'hôte du client configuré , et que la règle 125:8 est activée, le système génère une alerte et, dans le cadre d'un déploiement en ligne, abandonne les paquets incriminés. Ceci peut être utilisé pour empêcher les attaques par rebond FTP et permettre les connexions FTP lorsque la destination du canal de données FTP est différente de celle du client.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

ignore_telnet_erase_cmds

Indique si les séquences d'échappement Telnet relatives aux caractères d'effacement (TNC EAC) et au caractère d'effacement de ligne (TNC EAL) doivent être ignorées lors de la normalisation du canal de commande FTP. Ce paramètre doit correspondre à la façon dont le client FTP gère les commandes d'effacement Telnet. Généralement, les nouveaux clients FTP ignorent ces séquences d'échappement Telnet, tandis que les anciens clients les traitent. Lorsque le paramètre `ignore_telnet_erase_cmds` est défini sur `false`, l'inspecteur utilise la règle 125:1 pour générer des alertes et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

max_resp_len

Spécifie la longueur maximale de tous les messages de réponse acceptés par le client, en octets. Si le message d'une réponse FTP (tout ce qui se trouve après le code de retour à 3 chiffres) dépasse cette longueur et que la règle 125:6 est activée, le système génère une alerte et, dans le cadre d'un déploiement en ligne, abandonne les paquets incriminés. Ceci permet de rechercher les exploits de dépassement de tampon dans les clients FTP.

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `4 294 967 295`

telnet_cmds

Indique si la présence d'éventuelles commandes Telnet doit être recherchée sur le canal de commande FTP. Une telle présence peut révéler une tentative d'évasion sur ce canal.

Vous pouvez activer la règle 125:1 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

Règles de l'inspecteur de client FTP

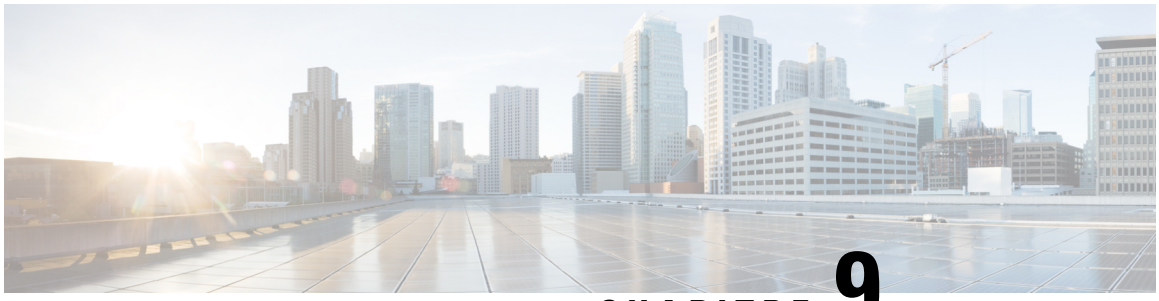
Activez les règles de l'inspecteur `ftp_client` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 9 : Règles de l'inspecteur de client FTP

GID:SID	Message de règle
125:1	Commande TELNET sur le canal de commande FTP
125:6	Message de réponse FTP trop long
125:8	Tentative de rebond FTP

Options des règles de prévention des intrusions de l'inspecteur de client FTP

L'inspecteur `ftp_client` ne comporte aucune option de règle de prévention des intrusions.



CHAPITRE 9

Inspecteur de serveur FTP

Présentation de l'inspecteur de serveur FTP

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`ftp_client`, `stream_tcp`
Activé	`vrai`

Le protocole FTP (File Transfer Protocol) est un protocole réseau utilisé pour transférer des fichiers entre des clients et des serveurs sur un réseau TCP/IP. Dès qu'une connexion est établie entre un client et un serveur, le client envoie des commandes au serveur pour charger ou télécharger des fichiers, et interprète les réponses reçues du serveur.

L'inspecteur `ftp_server` examine et normalise les commandes sur le canal de commande FTP.

À partir d'un tampon de canal de commande FTP, l'inspecteur `ftp_server` identifie les commandes et les paramètres FTP, et vérifie la validité de ces paramètres. L'inspecteur `ftp_server` détermine quand une connexion de commande FTP est chiffrée et quand un canal de données FTP est ouvert.

Paramètres de l'inspecteur de serveur FTP

Configuration des ports du serveur FTP

L'inspecteur `binder` configure le serveur FTP. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
    {
        "when": {
            "role":"any",
            "service":"ftp",
            "ports": ""
        },
        "use": {
            "type":"ftp_server"
        }
    }
]
```

chk_str_fmt

Spécifie une liste de commandes FTP à vérifier pour prévenir les attaques par format de chaîne. Vous pouvez activer la règle 125:5 pour générer une alerte et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés lorsque l'inspecteur détecte cette condition. Utilisez un espace comme séparateur entre les commandes.

Type : chaîne

Valeurs valides : liste de commandes FTP valides.

Valeur par défaut : aucune

data_chan_cmds

Spécifie une liste de commandes FTP dont la mise en forme doit être vérifiée. Utilisez un espace comme séparateur entre les commandes.

Type : chaîne

Valeurs valides : liste d'une ou plusieurs des commandes suivantes : `PORT PASV LPRT LPSV EPRT EPSV`.

Valeur par défaut : aucune

data_xfer_cmds

Spécifie une liste de commandes de transfert de données. Assurez-vous que la mise en forme des commandes est correcte. Utilisez un espace comme séparateur entre les commandes.

Type : chaîne

Valeurs valides : liste d'une ou plusieurs des commandes suivantes : `RETR STOR STOU APPE LIST NLST`.

Valeur par défaut : aucune

file_put_cmds

Spécifie une liste de commandes `PUT`. Assurez-vous que la mise en forme des commandes est correcte. Utilisez un espace comme séparateur entre les commandes.

Type : chaîne

Valeurs valides : liste d'une ou plusieurs des commandes suivantes : `STOR STOU APPE`.

Valeur par défaut : aucune

Mise en garde Ne modifiez pas le paramètre `file_put_cmds`, sauf si le service d'assistance vous le demande.

file_get_cmds

Spécifie une liste de commandes `GET`. Assurez-vous que la mise en forme des commandes est correcte. Utilisez un espace comme séparateur entre les commandes.

Type : chaîne

Valeurs valides : liste de commandes `GET`, telles que `RETR`.

Valeur par défaut : aucune

Mise en garde Ne modifiez pas le paramètre `file_get_cmds`, sauf si le service d'assistance vous le demande.

encr_cmds

Spécifie une liste de commandes relatives aux connexions sécurisées. Assurez-vous que la mise en forme des commandes est correcte. Utilisez un espace comme séparateur entre les commandes.

Type : chaîne

Valeurs valides : liste de commandes relatives aux connexions sécurisées, par exemple : `AUTH`.

Valeur par défaut : aucune

login_cmds

Spécifie une liste de commandes relatives au processus de connexion. Assurez-vous que la mise en forme des commandes est correcte. Utilisez un espace comme séparateur entre les commandes.

Type : chaîne

Valeurs valides : spécifiez une liste d'une ou plusieurs commandes : `USER`, `PASS`.

Valeur par défaut : aucune

check_encrypted

Indique si une session chiffrée doit être vérifiée pour détecter une commande visant à désactiver le chiffrement. À utiliser avec le paramètre `encrypted_traffic`.

En ce qui concerne ce paramètre, vous pouvez activer la règle 125:7 pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

cmd_validity[]

Tableau de commandes FTP et critères utilisés par l'inspecteur pour les valider. Ces contrôles de validité remplacent les contrôles par défaut effectués par l'inspecteur `ftp_server` (RFC 959).

En ce qui concerne ce paramètre, vous pouvez activer les règles 125:2 et 125:4 pour générer des événements et, dans le cadre d'un déploiement de ligne, abandonner les paquets incriminés.

Type : tableau (objet)

Exemple :

```
{
   "cmd_validity": [
       {
           "command": "CWD",
           "format": "abc",
           "length": 250
       }
   ]
}
```

cmd_validity[].command

Spécifie le nom d'une commande FTP à valider.

Type : chaîne

Valeurs valides : une commande FTP valide entre guillemets doubles.

Valeur par défaut : aucune

cmd_validity[].format

Décrit le format valide pour `cmd_validity[].command`

Type : chaîne

Valeurs valides : l'un des formats suivants :

- `int` : le paramètre doit être un entier.
- `number` : le paramètre doit être un entier compris entre 1 et 255.
- `char chars` : le paramètre doit être un caractère unique provenant de *`chars`* ou une liste d'un ou plusieurs caractères sans séparateurs.
- `date datefmt` : le paramètre suit le format spécifié, où *`datefmt`* est composé des éléments suivants :
  - `#` = Nombre
  - `C` = Caractère
  - `[]` = Format facultatif entre crochets
  - `|` = OU
  - `{}` = Choix de formats entre accolades
  - `.+-` = Caractères littéraux
- `string` : le paramètre est une chaîne illimitée.

- `host_port` : le paramètre doit être un spécificateur de port d'hôte, conformément à la RFC 959.
- `long_host_port` : le paramètre doit être un spécificateur de port d'hôte long, conformément à la RFC 1639.
- `extended_host_port` : le paramètre doit être un spécificateur de port d'hôte étendu, conformément à la RFC 2428.
- `{},|` : le paramètre doit être l'une des options entre accolades, séparées par |.
- `{}, []` : le paramètre doit être l'une des options entre accolades. Les valeurs facultatives sont placées entre crochets.

Valeur par défaut : aucune

cmd_validity[].length

Spécifie la longueur maximale en octets du paramètre `cmd_validity[].command`, en remplaçant la valeur par défaut définie dans `def_max_param_len`. Si le paramètre de la commande FTP dépasse la valeur de `cmd_validity[].length` et que la règle 125:3 est activée, Snort génère une alerte. Utilisez `cmd_validity[].length` pour imposer des restrictions sur la taille des valeurs de paramètres de commandes spécifiques.

Spécifiez `0` pour une longueur illimitée.

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `0`

def_max_param_len

Spécifie la longueur maximale par défaut, en octets, que l'inspecteur autorise pour toutes les commandes FTP gérées par le serveur. Utilisez `def_max_param_len` pour une détection basique de dépassement de tampon. (Cette valeur peut être redéfinie pour des commandes individuelles à l'aide de `cmd_validity[].length`.) En ce qui concerne ce paramètre, vous pouvez activer la règle 125:3 pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Spécifiez `0` pour une longueur illimitée.

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `100`

encrypted_traffic

Indique si le trafic FTP chiffré doit être vérifié. À utiliser avec le paramètre `check_encrypted`. En ce qui concerne ce paramètre, vous pouvez activer la règle 125:7 pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

ftp_cmds

Liste des commandes FTP prises en charge par le serveur en plus de celles décrites dans la RFC 959. (Si votre installation utilise les commandes « X » spécifiées dans la RFC 775, par exemple, vous pouvez les ajouter à l'inspecteur à l'aide de ce paramètre).

Type : chaîne

Valeurs valides : liste de commandes FTP valides, séparées par des espaces et entre guillemets doubles.

Valeur par défaut : aucune

ignore_data_chan

Indique si les canaux de données FTP doivent être ignorés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

ignore_telnet_erase_cmds

Indique si les séquences d'échappement Telnet relatives aux caractères d'effacement (TNC EAC) et au caractère d'effacement de ligne (TNC EAL) doivent être ignorées lors de la normalisation du canal de commande FTP. Définissez `ignore_telnet_erase_cmds` en tenant compte de la manière dont votre serveur FTP gère les commandes d'effacement Telnet. Généralement, les nouveaux clients FTP ignorent ces séquences d'échappement Telnet, tandis que les anciens clients les traitent.

Si les commandes d'effacement Telnet ne sont pas ignorées et que la règle 125:1 est activée, Snort génère un événement et, dans le cadre d'un déploiement en ligne, abandonne les paquets incriminés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

print_cmds

Indique si la configuration de chaque commande FTP relative à ce serveur doit être imprimée lors de l'initialisation.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

telnet_cmds

Indique si la présence d'éventuelles commandes Telnet doit être recherchée sur le canal de commande FTP. Une telle présence peut révéler une tentative d'évasion sur ce canal.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

Règles de l'inspecteur de serveur FTP

Activez les règles de l'inspecteur `ftp_server` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 10 : Règles de l'inspecteur de serveur FTP

GID:SID	Message de règle
125:1	Commande TELNET sur le canal de commande FTP
125:2	Commande FTP non valide
125:3	Paramètres de la commande FTP trop longs
125:4	Syntaxe des paramètres de la commande FTP incorrecte
125:5	Format de chaîne suspect dans les paramètres de la commande FTP
125:7	Trafic FTP chiffré
125:9	Commande TELNET évasive (incomplète) sur le canal de commande FTP

Options des règles de prévention des intrusions de l'inspecteur de serveur FTP

L'inspecteur `ftp_server` ne comporte aucune option de règle de prévention des intrusions.

CHAPITRE 10

Inspecteur GTP Inspect

Présentation de l'inspecteur GTP Inspect

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`stream_udp`
Activé	`faux`

Le protocole de tunnellisation GPRS (General Service Packet Radio) permet de communiquer sur un réseau central GTP.

L'inspecteur `gtp_inspect` détecte les anomalies au sein du trafic GTP et transfère les messages de signalisation du canal de commande au moteur de règles pour inspection.

Paramètres de l'inspecteur GTP Inspect

Configuration des ports et du service GTP Inspect

L'inspecteur `binder` configure les `ports` et le `service` GTP Inspect. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
   {
       "when": {
           "service": "gtp_inspect",
           "role": any
```

```
        },
        "use": {
            "type": "gtp_inspect"
        }
    },
    {
        "when": {
            "proto": "tcp",
            "role": "server",
            "ports": "2123 2152 3386"
        },
        "use": {
            "type": "gtp_inspect"
        }
    }
]
```

version

Spécifie une version GTP valide.

Type : entier

Valeurs valides : `0`, `1`, `2`

Valeur par défaut : `2`

messages[]

Spécifie un tableau d'informations sur les messages GTP valides.

Type : tableau (objet)

Exemple :

```
{
    messages: [
        {
            "type": 0,
            "name": ""
        }
    ]
}
```

messages[].type

Spécifie un type de message GTP valide. Voir le Tableau 12 : Types de messages GTP.

Type : entier

Plage valide : de `0` à `255`

Valeur par défaut : aucune

messages[].name

Spécifie un nom de message GTP valide. Voir le Tableau 12 : Types de messages GTP.

Type : chaîne

Valeurs valides : un nom de message GTP valide

Valeur par défaut : aucune

infos[]

Spécifie un tableau d'éléments d'information GTP.

Type : tableau (objet)

Exemple :

```
{
    infos: [
        {
            "type": 0,
            "name": "echo_request",
            "length": 0
        }
    ]
}
```

infos[].type

Spécifie un code de type d'élément GTP valide. Voir le Tableau 13 : Éléments d'information GTP.

Type : entier

Plage valide : de `0` à `255`

Valeur par défaut : `0`

infos[].name

Spécifie un nom d'élément GTP valide.

Type : chaîne

Valeurs valides : noms d'éléments d'information GTP valides. Voir le Tableau 13 : Éléments d'information GTP.

infos[].length

Spécifie la longueur d'un élément d'information GTP valide.

Type : entier

Plage valide : de `0` à `255`

Valeur par défaut : `0`

Règles de l'inspecteur GTP Inspect

Activez les règles de l'inspecteur `gtp_inspect` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 11 : Règles de l'inspecteur GTP

GID:SID	Message de règle
143:1	Longueur du message non valide
143:2	Longueur de l'élément d'information non valide

GID:SID	Message de règle
143:3	Éléments d'information non classés dans l'ordre
143:4	TEID manquant

Options des règles de prévention des intrusions de l'inspecteur GTP Inspect

Les options des règles de prévention des intrusions de l'inspecteur `gtp_inspect` vous permettent d'inspecter le canal de commande GTP pour analyser la version de GTP, le type de message et les éléments d'information.

Vous ne pouvez pas utiliser les options GTP en combinaison avec `content` ou `byte_jump`. Vous devez utiliser `gtp_version` dans chacune des règles qui utilisent `gtp_info` ou `gtp_type`.

gtp_version

Assurez-vous que la version de GTP spécifiée correspond à celle des messages de contrôle GTP.

Type : entier

Syntaxe : `gtp_version: <version>;`

Valeurs valides : `0`, `1`, `2`

Exemples : `gtp_version: 1;`

gtp_type

Chaque message GTP est identifié par un type de message, qui comprend une valeur numérique et une chaîne. Assurez-vous que chaque type de GTP spécifié correspond à celui des messages GTP.

Vous pouvez spécifier une valeur décimale définie pour un type de message, une chaîne définie ou une liste séparée par des virgules de l'un ou des deux, ou des deux, dans n'importe quelle combinaison, comme le montre l'exemple suivant :

Type : chaîne

Syntaxe : `gtp_type: <message_type>;`

Valeurs valides : répertoriées dans le tableau des types de messages GTP. Voir le Tableau 12 : Types de messages GTP.

Exemples : `gtp_type: "10, 11, echo_request";`

Le système utilise une opération OU pour mettre en correspondance chaque valeur ou chaîne que vous répertoriez. L'ordre dans lequel vous répertoriez les valeurs et les chaînes n'a pas d'importance. Toute valeur ou chaîne unique de la liste correspond au mot-clé. Le système génère une erreur si vous tentez d'enregistrer une règle qui comprend une chaîne non reconnue ou une valeur hors limites.

Notez que les différentes versions de GTP utilisent parfois des valeurs différentes pour le même type de message. Par exemple, la valeur du type de message `sgsn_context_request` est de `50` dans GTPv0 et GTPv1, et de`130` dans GTPv2.

L'option `gtp_type` peut prendre différentes valeurs selon le numéro de version indiqué dans le paquet. Par exemple, la valeur du message `sgsn_context_request` est de `50` dans un paquet GTPv0 ou GTPv1, et de `130` dans un paquet GTPv2. Si la valeur du type de message ne correspond pas à une valeur connue pour la version indiquée dans le paquet, l'option ne trouve pas de correspondance.

Lorsque vous spécifiez un entier pour le type de message, l'option recherche une correspondance si le type de message correspond à la valeur du paquet GTP, indépendamment de la version indiquée dans le paquet.

`gtp_message_type` est une valeur numérique ou un mot clé provenant du Tableau 12 : Types de messages GTP.

Tableau 12 : Types de messages GTP

Type	Nom pour la version 0	Nom pour la version 1	Nom pour la version 2
1	echo_request	echo_request	echo_request
2	echo_response	echo_response	echo_response
3	version_not_supported	version_not_supported	version_not_supported
4	node_alive_request	node_alive_request	S.O.
5	node_alive_response	node_alive_response	S.O.
6	redirection_request	redirection_request	S. O.
7	redirection_response	réponse_redirection	S. O.
16	create_pdp_context_request	create_pdp_context_request	S. O.
17	create_pdp_context_response	create_pdp_context_response	S. O.
18	update_pdp_context_request	update_pdp_context_request	S. O.
19	update_pdp_context_response	update_pdp_context_response	S. O.
20	delete_pdp_context_request	delete_pdp_context_request	S. O.
21	delete_pdp_context_response	delete_pdp_context_response	S. O.
22	create_aa_pdp_context_request	init_pdp_context_activation_request	S.O.
23	create_aa_pdp_context_response	init_pdp_context_activation_response	S. O.
24	delete_aa_pdp_context_request	s.o.	s.o.
25	delete_aa_pdp_context_response	s.o.	s.o.
26	error_indication	error_indication	S. O.
27	pdu_notification_request	pdu_notification_request	S. O.
28	pdu_notification_response	pdu_notification_response	S. O.
29	pdu_notification_reject_request	pdu_notification_reject_request	S. O.

Type	Nom pour la version 0	Nom pour la version 1	Nom pour la version 2
30	pdu_notification_reject_response	pdu_notification_reject_response	S. O.
31	S. O.	supported_ext_header_notification	S. O.
32	send_routing_info_request	send_routing_info_request	create_session_request
33	send_routing_info_response	send_routing_info_response	create_session_response
34	failure_report_request	failure_report_request	modify_bearer_request
35	failure_report_response	failure_report_response	modify_bearer_response
36	note_ms_present_request	note_ms_présent_request	delete_session_request
37	note_ms_present_response	note_ms_present_response	delete_session_response
38	s.o.	s.o.	change_notification_request
39	s.o.	s.o.	change_notification_response
48	identification_request	identification_request	S. O.
49	identification_response	identification_response	S. O.
50	sgsn_context_request	sgsn_context_request	S. O.
51	sgsn_context_response	sgsn_context_response	S. O.
52	sgsn_context_ack	sgsn_context_ack	S. O.
53	s.o.	forward_relocation_request	S. O.
54	s.o.	forward_relocation_response	S. O.
55	s.o.	forward_relocation_complete	S.O.
56	s.o.	relocation_cancel_request	S. O.
57	s.o.	relocation_cancel_response	S. O.
58	S. O.	forward_srns_contex	S. O.
59	s.o.	forward_relocation_complete_ack	S. O.
60	S. O.	forward_srns_contex_ack	S. O.
64	s.o.	s.o.	modify_bearer_command
65	s.o.	s.o.	modify_bearer_failure_indication
66	s.o.	s.o.	delete_bearer_command
67	s.o.	s.o.	delete_bearer_failure_indication
68	s.o.	s.o.	bearer_resource_command

Type	Nom pour la version 0	Nom pour la version 1	Nom pour la version 2
69	s.o.	s.o.	Bearer_resource_failure_indication
70	s.o.	ran_info_relay	descendant_failure_indication
71	s.o.	s.o.	trace_session_activation
72	s.o.	s.o.	trace_session_deactivation
73	s.o.	s.o.	stop_pages_indication
95	s.o.	s.o.	create_bearer_request
96	S. O.	mbms_notification_request	create_bearer_response
97	S. O.	mbms_notification_response	update_bearer_request
98	S. O.	mbms_notification_reject_request	Update_bearer_response
99	S. O.	mbms_notification_reject_response	delete_bearer_request
100	S. O.	create_mbms_context_request	delete_bearer_response
101	S. O.	create_mbms_context_response	delete_pdn_request
102	S. O.	update_mbms_context_request	delete_pdn_response
103	S. O.	Update_mbms_context_response	S. O.
104	S. O.	delete_mbms_context_request	S. O.
105	S. O.	delete_mbms_context_response	S. O.
112	S. O.	mbms_register_request	S. O.
113	S. O.	mbms_register_response	S. O.
114	S. O.	mbms_deregister_request	S. O.
115	S. O.	mbms_deregister_response	S. O.
116	s.o.	mbms_session_start_request	S. O.
117	s.o.	mbms_session_start_response	S. O.
118	s.o.	mbms_session_stop_request	S. O.
119	s.o.	mbms_session_stop_response	S. O.
120	s.o.	mbms_session_update_request	S. O.
121	s.o.	mbms_session_update_response	S. O.
128	S. O.	ms_info_change_request	identification_request
129	S. O.	ms_info_change_response	identification_response

Type	Nom pour la version 0	Nom pour la version 1	Nom pour la version 2
130	S.O.	S.O.	sgsn_context_request
131	S.O.	S.O.	sgsn_context_response
132	S.O.	S.O.	sgsn_context_ack
133	S.O.	S.O.	forward_relocation_request
134	S.O.	S.O.	forward_relocation_response
135	S.O.	S.O.	forward_relocation_complete
136	S.O.	S.O.	forward_relocation_complete_ack
137	S.O.	S.O.	forward_access
138	S.O.	S.O.	forward_access_ack
139	S.O.	S.O.	relocation_cancel_request
140	S.O.	S.O.	relocation_cancel_response
141	S.O.	S.O.	configuration_transfer_tunnel
149	S.O.	S.O.	dissocier
150	S.O.	S.O.	detach_ack
151	S.O.	S.O.	cs_paging
152	S.O.	S.O.	ran_info_relay
153	S.O.	S.O.	alerte_mme
154	S.O.	S.O.	alert_mme_ack
155	S.O.	S.O.	ue_activity
156	S.O.	S.O.	ue_activity_ack
160	S.O.	S.O.	create_forward_tunnel_request
161	S.O.	S.O.	create_forward_tunnel_response
162	S.O.	S.O.	suspend
163	S.O.	S.O.	suspend_ack
164	S.O.	S.O.	reprendre
165	S.O.	S.O.	resume_ack
166	S.O.	S.O.	create_indirect_forward_tunnel_request
167	S.O.	S.O.	create_indirect_forward_tunnel_response

Type	Nom pour la version 0	Nom pour la version 1	Nom pour la version 2
168	S.O.	S.O.	delete_indirect_forward_tunnel_request
169	S.O.	S.O.	delete_indirect_forward_tunnel_response
170	S.O.	S.O.	Release_access_bearer_request
171	S.O.	S.O.	Release_access_bearer_response
176	S.O.	S.O.	downlink_data
177	S.O.	S.O.	download_data_ack
179	S.O.	S.O.	pgw_restart
180	S.O.	S.O.	pgw_restart_ack
200	S.O.	S.O.	Update_pdn_request
201	S.O.	S.O.	update_pdn_response
211	S.O.	S.O.	modify_access_bearer_request
212	S.O.	S.O.	modify_access_bearer_response
231	S.O.	S.O.	mbms_session_start_request
232	S.O.	S.O.	mbms_session_start_response
233	S.O.	S.O.	mbms_session_update_request
234	S.O.	S.O.	mbms_session_update_response
235.	S.O.	S.O.	mbms_session_stop_request
236	S.O.	S.O.	mbms_session_stop_response
240	data_record_transfer_request	data_record_transfer_request	S. O.
241	data_record_transfer_response	data_record_transfer_response	S. O.
254	S. O.	end_marker	S. O.
255	pdu	pdu	S.O.

gtp_info

Un message GTP peut inclure plusieurs éléments d'information, chacun étant identifié à la fois par une valeur numérique et une chaîne définies. Vous pouvez utiliser l'option `gtp_info` pour commencer l'inspection au début d'un élément d'information spécifié et limiter l'inspection à celui-ci.

Vous pouvez spécifier la valeur décimale définie ou la chaîne définie pour un élément d'information. Vous pouvez spécifier une valeur ou une chaîne unique, et vous pouvez utiliser plusieurs options `gtp_info` au sein d'une règle pour inspecter plusieurs éléments d'information.

Lorsqu'un message comprend plusieurs éléments d'information du même type, tous sont examinés pour vérifier s'ils correspondent. Lorsque des éléments d'information apparaissent dans un ordre non valide, seule la dernière instance est inspectée.

Selon la version, un message GTP peut utiliser différentes valeurs pour le même élément d'information. Par exemple, la valeur de l'élément d'information `cause` est de `1` dans GTPv0 et GTPv1, et de `2` dans GTPv2.

L'option `gtp_info` peut prendre différentes valeurs selon le numéro de version indiqué dans le paquet. Dans l'exemple ci-dessus, pour le mot clé, la valeur de l'élément d'information est de `1` dans un paquet GTPv0 ou GTPv1, et de `2` dans un paquet GTPv2. Si la valeur de l'élément d'information ne correspond pas à une valeur connue pour la version spécifiée dans le paquet, l'option ne recherche pas de correspondance.

Lorsque vous spécifiez un entier pour l'élément d'information, l'option recherche une correspondance si le type de message correspond à la valeur du paquet GTP, indépendamment de la version indiquée dans le paquet.

Type : chaîne

Syntaxe : `gtp_info: <identifier>;`

Valeurs valides : répertoriées dans le Tableau 13 : Éléments d'information GTP.

Exemples : `gtp_info: "qos";`

Tableau 13 : Éléments d'information GTP

Type	Nom pour la version 0	Nom pour la version 1	Nom pour la version 2
1	cause	cause	imsi
2	imsi	imsi	cause
3	rai	rai	recovery
4	tlli	tlli	S.O.
5	p_tmsi	p_tmsi	S.O.
6	qos	s.o.	s.o.
8	recording_required	recording_required	S.O.
9	authentication	authentication	S.O.
10	s.o.	s.o.	s.o.
11	map_cause	map_cause	S. O.
12	p_tmsi_sig	p_tmsi_sig	S. O.
13	ms_validated	ms_validated	S.O.
14	recovery	recovery	S.O.
15	selection_mode	selection_mode	S. O.
16	flow_label_data_1	teid_1	S. O.
17	flow_label_signalling	teid_control	S. O.

Type	Nom pour la version 0	Nom pour la version 1	Nom pour la version 2
18	flow_label_data_2	teid_2	S. O.
19	ms_unreachable	teardown_ind	S. O.
20	S. O.	nsapi	S. O.
21	S. O.	ranap	S. O.
22	S. O.	rab_context	S.O.
23	S. O.	radio_priority_sms	S. O.
24	S. O.	radio_priority	s.o.
25	s.o.	packet_flow_id	s.o.
26	s.o.	charging_char	S. O.
27	s.o.	trace_ref	S. O.
28	S. O.	trace_type	S. O.
29	S. O.	ms_unreachable	S. O.
71	s.o.	s.o.	apn
72	s.o.	s.o.	ambr
73	s.o.	s.o.	ebi
74	s.o.	s.o.	ip_addr
75	s.o.	s.o.	mei
76	s.o.	s.o.	msisdn
77	s.o.	s.o.	Indication
78	s.o.	s.o.	pco
79	s.o.	s.o.	paa
80	s.o.	s.o.	bearer_qos
80	s.o.	s.o.	flow_qos
82	s.o.	s.o.	rat_type
83	s.o.	s.o.	serving_network
84	s.o.	s.o.	bearer_tft
85	s.o.	s.o.	tad
86	s.o.	s.o.	uli

Type	Nom pour la version 0	Nom pour la version 1	Nom pour la version 2
87	S.O.	S.O.	f_teid
88	S.O.	S.O.	tmsi
89	S.O.	S.O.	cn_id
90	S.O.	S.O.	s103pdf
91	S.O.	S.O.	s1udf
92	S.O.	S.O.	delay_value
93	S.O.	S.O.	bearer_context
94	S.O.	S.O.	charging_id
95	S.O.	S.O.	charging_char
96	S.O.	S.O.	trace_info
97	S.O.	S.O.	bearer_flag
99	S.O.	S.O.	pdn_type
100	S.O.	S.O.	pti
101	S.O.	S.O.	drx_parameter
103	S.O.	S.O.	gsm_key_tri
104	S.O.	S.O.	umts_key_cipher_quin
105	S.O.	S.O.	gsm_key_cipher_quin
106	S.O.	S.O.	umts_key_quin
107	S.O.	S.O.	eps_quad
108	S.O.	S.O.	umts_key_quad_quin
109	S.O.	S.O.	pdn_connection
110	S.O.	S.O.	pdn_number
111	S.O.	S.O.	p_tmsi
112	S.O.	S.O.	p_tmsi_sig
113	S.O.	S.O.	hop_counter
114	S.O.	S.O.	ue_time_zone
115	S.O.	S.O.	trace_ref
116	S.O.	S.O.	complete_request_msg

Type	Nom pour la version 0	Nom pour la version 1	Nom pour la version 2
117	S.O.	S.O.	guti
118	S.O.	S.O.	f_container
119	S.O.	S.O.	f_cause
120	S.O.	S.O.	plmn_id
121	S.O.	S.O.	target_id
123	S.O.	S.O.	packet_flow_id
124	S.O.	S.O.	rab_contex
125	S.O.	S.O.	src_rnc_pdcp
126	S.O.	S.O.	udp_src_port
127	charge_id	charge_id	apn_restriction
128	end_user_address	end_user_address	selection_mode
129	mm_context	mm_context	src_id
130	pdp_context	pdp_context	S. O.
131	apn	apn	change_report_action
132	protocol_config	protocol_config	fq_csid
133	gsn	gsn	channel
134	msisdn	msisdn	emlpp_pri
135	S.O.	qos	node_type
136	S.O.	authentication_qu	fqdn
137	S.O.	tft	ti
138	S.O.	target_id	mbms_session_duration
139	S.O.	utran_trans	mbms_service_area
140	S.O.	rab_setup	mbms_session_id
141	S.O.	ext_header	mbms_flow_id
142	S. O.	trigger_id	mbms_ip_multicast
143	S. O.	omc_id	mbms_distribution_ack
144	S. O.	ran_trans	rfsp_index
145	S. O.	pdp_context_pri	uci

Type	Nom pour la version 0	Nom pour la version 1	Nom pour la version 2
146	S. O.	addi_rab_setup	csg_info
147	S. O.	sgsn_number	csg_id
148	S. O.	common_flag	cmi
149	s.o.	apn_restriction	service_indicator
150	s.o.	radio_priority_lcs	detach_type
151	s.o.	rat_type	ldn
152	s.o.	user_loc_info	node_feature
153	s.o.	ms_time_zone	mbms_time_to_transfer
154	s.o.	imei_sv	throttling
155	s.o.	camel	arp
156	s.o.	mbms_ue_context	epc_timer
157	S. O.	tmp_mobile_group_id	signalling_priority_indication
158	S. O.	rim_routing_addr	tmgi
159	S. O.	mbms_config	mm_srvcc
160	s.o.	mbms_service_area	flags_srvcc
161	s.o.	src_rnc_pdcp	nmbr
162	s.o.	addi_trace_info	S. O.
163	s.o.	hop_counter	S. O.
164	s.o.	plmn_id	S. O.
165	s.o.	mbms_session_id	S. O.
166	s.o.	mbms_2g3g_indicator	S. O.
167	s.o.	enhanced_nsapi	S. O.
168	s.o.	mbms_session_duration	S. O.
169	s.o.	addi_mbms_trace_info	S. O.
170	s.o.	mbms_session_repetition_num	S. O.
171	s.o.	mbms_time_to_data	S. O.
173	S. O.	bss	S. O.
174	S. O.	cell_id	S. O.

Type	Nom pour la version 0	Nom pour la version 1	Nom pour la version 2
175	S. O.	pdu_num	S. O.
177	s.o.	mbms_bearer_capab	S. O.
178	S. O.	rim_routing_disc	S. O.
179	s.o.	list_pfc	S. O.
180	s.o.	ps_xid	S. O.
181	S. O.	ms_info_change_report	S. O.
182	S. O.	direct_tunnel_flags	S. O.
183	S. O.	correlation_id	S. O.
184	S. O.	bearer_control_mode	S. O.
185	S. O.	mbms_flow_id	S. O.
186	S. O.	mbms_ip_multicast	S. O.
187	S. O.	mbms_distribution_ack	S. O.
188	S. O.	reliable_inter_rat_handover	S. O.
189	S. O.	rfsp_index	S. O.
190	S. O.	fqdn	S. O.
191	S. O.	evolved_allocation1	S. O.
192	S. O.	evolved_allocation2	S. O.
193	S. O.	extended_flags	S. O.
194	S. O.	uci	S. O.
195	S. O.	csg_info	S. O.
196	S. O.	csg_id	S. O.
197	S. O.	cmi	S. O.
198	S. O.	apn_ambr	S. O.
199	S. O.	ue_network	S. O.
200	s.o.	ue_ambr	S. O.
201	s.o.	apn_ambr_nsapi	S. O.
202	S. O.	ggsn_backoff_timer	S. O.
203	S. O.	signalling_priority_indication	S. O.

Type	Nom pour la version 0	Nom pour la version 1	Nom pour la version 2
204	S. O.	signalling_priority_indication_nsapi	S. O.
205	S. O.	high_bitrate	S. O.
206	S. O.	max_mbr	S. O.
251	charging_gateway_addr	charging_gateway_addr	S. O.
255	private_extension	private_extension	private_extension

CHAPITRE 11

Inspecteur HTTP Inspect

Présentation de l'inspecteur HTTP Inspect

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`stream_tcp`
Activé	`vrai`

Le protocole HTTP (Hypertext Transfer Protocol) est un protocole de couche application qui permet l'échange d'hypermédias (audio, vidéo, images et texte) entre un client et un serveur. HTTP est un protocole sans état qui nécessite une transmission fiable des messages. La communication client-serveur s'effectue au moyen de demandes et de réponses HTTP.

Un serveur HTTP/1.1 utilise généralement le port 80 sur TCP/IP. La version sécurisée de HTTP (HTTP/TLS ou HTTPS) utilise le port 443. Le protocole HTTP intègre des mécanismes de contrôle d'accès et d'authentification.

HTTP/2 contient des améliorations visant à accroître la vitesse et à transmettre plus d'informations que le client n'en a demandées, mais fonctionne sur les mêmes ports et protocoles que HTTP/1.1. Les règles spécifiques à HTTP/2 sont configurées avec `service:http2`.

HTTP/3 est sans connexion, utilise le protocole QUIC (Quick UDP Internet Connections) plutôt que TCP, et peut prendre en charge davantage de flux actifs avec une meilleure récupération des pertes. HTTP/3 utilise le même système de messagerie que les versions précédentes de HTTP. Les règles spécifiques à HTTP/3 sont configurées avec `service:http3`.

L'inspecteur HTTP prend en charge les trois versions de HTTP de la même manière.

L'inspecteur `http_inspect` détecte et analyse l'unité de données de protocole (PDU) du message HTTP. `http_inspect` reçoit la charge utile TCP du flux TCP et examine le message HTTP encapsulé.

L'inspecteur HTTP peut détecter les sections suivantes des messages HTTP :

- Ligne de demande
- Ligne d'état
- En-têtes
- Corps du message Content-Length (corps du message défini par l'en-tête Content-Length)
- Corps du message segmenté
- Corps du message précédent (corps du message sans en-tête Content-Length)
- « Trailers »

L'inspecteur `http_inspect` détecte et normalise tous les champs d'en-tête HTTP et les composants de l'URI HTTP. L'inspecteur `http_inspect` ne normalise pas le port TCP.

L'inspecteur `http_inspect` peut détecter quatre types d'URI :

- Astérisque (`*`) : non normalisé
- Autorité : URI utilisé avec la méthode HTTP CONNECT
- Origine : URI qui commence par une barre oblique (aucun schéma ou autorité présent)
- Absolu : URI qui comprend un schéma, un hôte et un chemin absolu

Un URI HTTP peut inclure ce qui suit :

- Schéma (ftp, http ou https)
- Hôte (nom de domaine du serveur)
- Port TCP
- Chemin (répertoire et fichier)
- Requête (paramètres de la demande)
- Fragment (portion du fichier)

Vous pouvez configurer l'inspecteur `http_inspect` pour qu'il génère des alertes sur les sections du message HTTP. Par exemple :

- Spécifier le nombre d'octets à lire dans le corps de la demande ou de la réponse HTTP
- Activer la détection et la normalisation de JavaScript
- Gérer différents types de décompression de fichiers
- Personnaliser le décodage de l'URI HTTP

Remarque `http_inspect` peut effectuer une inspection partielle de la charge utile du flux TCP.

Bonnes pratiques en matière de configuration de l'inspecteur HTTP Inspect

Tenez compte des bonnes pratiques suivantes lors de la configuration de l'inspecteur `http_inspect` :

- Définissez les paramètres `request_ depth` et `response_ depth` si votre trafic HTTP comprend des fichiers vidéo volumineux.
- Utilisez les paramètres par défaut pour l'inspection des URI HTTP :

```
"utf8": "true"
"plus_to_space": "true"
"percent_u": "true"
"utf8_bare_byte": "true"
"iis_unicode": "true"
"iis_double_decode": "true"
```

Paramètres de l'inspecteur HTTP Inspect

Configuration du service HTTP

L'inspecteur `binder` configure le service HTTP. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
    {
        "when": {
             "service": "http",
             "role": any
        },
        "use": {
            "type": "http_inspect"
        }
    }
]
```

request_depth

Spécifie le nombre d'octets à lire dans le corps du message de demande HTTP.

Spécifiez `-1` pour n'appliquer aucune limite au nombre d'octets à inspecter. Nous vous recommandons de spécifier les paramètres `request_depth` et `response_depth` pour limiter le volume de données du corps HTTP à analyser.

Pour n'inspecter que les en-têtes HTTP, définissez `request_ depth` sur `0`.

Type : entier

Plage valide : de `-1` à `9 007 199 254 740 992 (max53)`

Valeur par défaut : `-1`

response_depth

Spécifie le nombre d'octets à lire dans le corps du message de réponse HTTP.

Spécifiez `-1` pour n'appliquer aucune limite au nombre d'octets à inspecter. Nous vous recommandons de spécifier les paramètres `request_depth` et `response_depth` pour limiter le volume de données du corps HTTP à analyser.

Pour n'inspecter que les en-têtes HTTP, définissez `response_depth` sur `0`.

Type : entier

Plage valide : de `-1` à `9 007 199 254 740 992 (max53)`

Valeur par défaut : `-1`

unzip

Indique si les fichiers `gzip` et le corps des messages doivent être décompressés avant de les inspecter. Lorsque vous désactivez la décompression, l'inspecteur HTTP ne peut pas traiter toutes les parties du corps du message HTTP. L'inspecteur `http_inspect` peut traiter les en-têtes HTTP.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut : `true`

maximum_host_length

Spécifie le nombre maximal d'octets autorisés dans la valeur d'en-tête HTTP `Host`.

Spécifiez `-1` pour ne pas limiter la longueur de la valeur d'en-tête.

Type : entier

Plage valide : de `-1` à `9 007 199 254 740 992 (max53)`

Valeur par défaut : `-1`

maximum_chunk_length

Spécifie le nombre maximal d'octets autorisés dans un bloc de corps de message HTTP.

Spécifiez `-1` pour n'appliquer aucune limite au nombre d'octets d'un bloc HTTP.

Type : entier

Plage valide : de `-1` à `9 007 199 254 740 992 (max53)`

Valeur par défaut : `-1`

normalize_utf

Indique si les encodages UTF (UTF-8, UTF-7, UTF-16LE, UTF-16BE, UTF-32LE et UTF-32BE) trouvés dans le corps de la réponse HTTP doivent être normalisés. L'inspecteur `http_inspect` détermine l'encodage des caractères UTF à partir de l'en-tête HTTP `Content-Type`.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut : `true`

decompress_pdf

Indique si les parties compressées au format deflate des fichiers `application/pdf` (PDF) trouvés dans le corps de la réponse HTTP doivent être décompressés. L'inspecteur `http_inspect` décompresse les fichiers PDF à l'aide du filtre de flux `/FlateDecode`.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut : `false`

decompress_swf

Indique si les fichiers `application/vnd.adobe.flash-movie` (SWF) trouvés dans le corps de la réponse HTTP doivent être décompressés.

Remarque Seules les parties compressées des fichiers trouvés dans les réponses HTTP GET peuvent être décompressées.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut :`false`

decompress_vba

Indique si les fichiers de macros Microsoft Office Visual Basic for Applications trouvés dans le corps de la réponse HTTP doivent être décrompressés.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut : `false`

decompress_zip

Indique si les fichiers `application/zip` (ZIP) trouvés dans le corps de la réponse HTTP doivent être décompressés.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut : `false`

script_detection

Indique si le contenu JavaScript doit être inspecté après la détection de l'élément de fin de script (<\script>). Lorsque `http_inspect` détecte la fin d'un script, il transmet immédiatement le corps du message partiellement lu pour permettre une détection précoce. La détection des scripts permet à Snort de bloquer rapidement les messages de réponse susceptibles de contenir du code JavaScript malveillant.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut :`false`

normalize_javascript

Indique si le mécanisme existant doit être utilisé pour normaliser JavaScript dans le corps de la réponse HTTP. Cette option configure le normalisateur JavaScript existant. L'inspecteur `http_inspect` normalise les données JavaScript obscurcies, y compris les fonctions `unescape` et `decodeURI`, et la méthode `String.fromCharCode`. L'inspecteur HTTP normalise les encodages dans les fonctions unescape, decodeURI et decodeURIComponent : %XX, %uXXXX, XX et uXXXXi.

L'inspecteur `http_inspect` détecte les espaces consécutifs et les normalise en un seul espace. Lorsque `normalize_javascript` est activé, vous pouvez définir `max_javascript_whitespaces` pour limiter le nombre d'espaces consécutifs dans les données Javascript obscurcies.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut : `false`

js_norm_bytes_depth

Spécifie le nombre d'octets JavaScript d'entrée à normaliser. Cette option est propre au normalisateur JavaScript amélioré.

Remarque Si vous utilisez le normalisateur JavaScript amélioré, les paramètres par défaut du LSP et de Snort 3 sont utilisés. Les configurations spécifiques à JavaScript sont bloquées dans l'interface utilisateur de la politique d'analyse de réseau (NAP). Pour remplacer les paramètres par défaut et personnaliser les paramètres du normalisateur, vous pouvez modifier le fichier `NAPOverride.lua` situé sous `/ftd/app_data/Volume/root1/ngfw/var/cisco/deploy`.

L'inspecteur `http_inspect` détecte les espaces consécutifs et les normalise en un seul espace. L'inspecteur assure le suivi des scripts répartis sur différentes PDU, où la balise de début <script> is in one PDU and the end </script> se trouve dans une PDU et la balise de fin dans une autre pour une normalisation efficace du trafic. Un nouveau tampon `js_data` a été ajouté au tampon IPS de Snort 3. Celui-ci utilise l'approche « Juste à temps » pour détecter et normaliser le code JavaScript, le normalisateur n'étant appelé que lorsque cette option est utilisée dans la règle.

L'inspecteur `http_inspect` normalise le nom de la fonction, le nom de la variable et le nom de l'étiquette associés au code JavaScript. En outre, l'inspecteur normalise le code JavaScript transféré sous forme de script externe en utilisant le type MIME application/javascript ou un type similaire. Le normalisateur insère automatiquement des points-virgules lorsque la fonctionnalité JavaScript n'est pas modifiée par rapport à l'entrée d'origine côté client.

L'inspecteur `http_inspect` normalise les données JavaScript obscurcies, y compris les fonctions `unescape`, `decodeURI` et `decodeURIComponent`, ainsi que les méthodes `String.fromCharCode` et `String.fromCodePoint`. L'inspecteur HTTP normalise les encodages dans les fonctions unescape, decodeURI et decodeURIComposant : %XX, %uXXXX, \uXX, \u{XXXX}\xXX, le point de code décimal et le point de code hexadécimal.

L'inspecteur `http_inspect` normalise également l'opérateur Javascript plus (+) et concatène les chaînes à l'aide de cet opérateur.

Spécifiez `-1` pour n'appliquer aucune limite au nombre d'octets JavaScript.

Type : entier

Plage valide : de `-1` à `9 007 199 254 740 992 (max53)`

Valeur par défaut : `-1`

js_norm_identifier_depth

Spécifie le nombre maximal d'identifiants JavaScript uniques à normaliser. Cette option est propre au normalisateur JavaScript amélioré.

Remarque Si vous utilisez le normalisateur JavaScript amélioré, les paramètres par défaut du LSP et de Snort 3 sont utilisés. Les configurations spécifiques à JavaScript sont bloquées dans l'interface utilisateur de la politique d'analyse de réseau (NAP). Pour remplacer les paramètres par défaut et personnaliser les paramètres du normalisateur, vous pouvez modifier le fichier `NAPOverride.lua` situé sous `/ftd/app_data/Volume/root1/ngfw/var/cisco/deploy`.

Type : entier

Plage valide : de `0` à `65 536`

Valeur par défaut : `65 536`

js_norm_max_bracket_depth

Spécifie la profondeur maximale de l'imbrication des crochets JavaScript à normaliser. Cette option est propre au normalisateur JavaScript amélioré.

Remarque Si vous utilisez le normalisateur JavaScript amélioré, les paramètres par défaut du LSP et de Snort 3 sont utilisés. Les configurations spécifiques à JavaScript sont bloquées dans l'interface utilisateur de la politique d'analyse de réseau (NAP). Pour remplacer les paramètres par défaut et personnaliser les paramètres du normalisateur, vous pouvez modifier le fichier `NAPOverride.lua` situé sous `/ftd/app_data/Volume/root1/ngfw/var/cisco/deploy`.

Type : entier

Plage valide : de `1` à `65 535`

Valeur par défaut : `256`

js_norm_max_scope_depth

Spécifie la profondeur maximale de l'imbrication de la portée JavaScript à normaliser. Cette option est propre au normalisateur JavaScript amélioré.

Remarque Si vous utilisez le normalisateur JavaScript amélioré, les paramètres par défaut du LSP et de Snort 3 sont utilisés. Les configurations spécifiques à JavaScript sont bloquées dans l'interface utilisateur de la politique d'analyse de réseau (NAP). Pour remplacer les paramètres par défaut et personnaliser les paramètres du normalisateur, vous pouvez modifier le fichier `NAPOverride.lua` situé sous `/ftd/app_data/Volume/root1/ngfw/var/cisco/deploy`.

Type : entier

Plage valide : de `1` à `65 535`

Valeur par défaut : `256`

js_norm_max_tmpl_nest

Spécifie la profondeur maximale de l'imbrication littérale du modèle JavaScript à normaliser. Cette option est propre au normalisateur JavaScript amélioré.

Remarque Si vous utilisez le normalisateur JavaScript amélioré, les paramètres par défaut du LSP et de Snort 3 sont utilisés. Les configurations spécifiques à JavaScript sont bloquées dans l'interface utilisateur de la politique d'analyse de réseau (NAP). Pour remplacer les paramètres par défaut et personnaliser les paramètres du normalisateur, vous pouvez modifier le fichier `NAPOverride.lua` situé sous `/ftd/app_data/Volume/root1/ngfw/var/cisco/deploy`.

Type : entier

Plage valide : de `0` à `255`

Valeur par défaut : `32`

max_javascript_whitespaces

Spécifie le nombre maximal d'espaces consécutifs autorisés dans les données JavaScript obscurcies.

Type : entier

Plage valide : de `1` à `65 535`

Valeur par défaut : `200`

percent_u

Indique si les encodages `%uNNNN` et `%UNNNN` doivent être normalisés. Les quatre caractères `N` représentent une valeur encodée en hexadécimal qui correspond à un point de code Unicode de Microsoft Internet Information Services (IIS). Comme les clients légitimes utilisent rarement les encodages `%u`, nous vous recommandons de normaliser le trafic HTTP codé avec des encodages `%u`.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut : `false`

utf8

Indique si les séquences Unicode UTF-8 standard doivent être normalisées dans l'URI. L'inspecteur `http_inspect` peut normaliser des caractères UTF-8 de deux ou trois octets en un seul octet.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut : `true`

utf8_bare_byte

Indique si les caractères UTF-8 incluant des octets qui ne sont pas codés en URL ou en pourcentage doivent être normalisés. Nous vous recommandons d'activer le paramètre `utf8_bare_byte`.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut : `false`

iis_unicode

Indique si les caractères du message HTTP doivent être normalisés en utilisant leur point de code Unicode.

Remarque Nous vous recommandons d'activer le paramètre `iis_unicode`. Les attaques et les tentatives d'évasion ont souvent recours à Unicode.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut :`false`

iis_unicode_code_page

Indique si la page de code du fichier de correspondance Unicode IIS doit être utilisée.

Type : entier

Plage valide : de `1` à `65 535`

Valeur par défaut : `1 252`

iis_double_decode

Indique si les caractères doivent être normalisés en effectuant un double décodage des caractères encodés en URL. Décode le trafic IIS doublement encodé en effectuant deux passages dans l'URI de demande. Nous vous recommandons d'activer le paramètre `iis_double_decode`. Le double encodage n'est généralement observé que dans les scénarios d'attaque.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut : `true`

oversize_dir_length

Spécifie le nombre maximal d'octets autorisés pour le répertoire de l'URL.

Type : entier

Plage valide : de `1` à `65 535`

Valeur par défaut : `300`

backslash_to_slash

Indique si la barre oblique inverse (`\`) doit être remplacée par une barre oblique (`/`) dans les URI.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut : `true`

plus_to_space

Indique si le signe plus (+) doit être remplacé par <sp> dans les URI.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut : `true`

simplify_path

Indique si le chemin d'accès au répertoire de l'URI doit être réduit à sa forme la plus simple. Les chemins complexes peuvent contenir les caractères suivants : `.`, `..` et `/`.

Type : booléen

Valeurs valides : `true, false`

Valeur par défaut : `true`

xff_headers

Spécifie les types d'en-tête HTTP `X-Forwarded-For` à examiner. Dans le paramètre `xff_headers`, dressez la liste des en-têtes `X-Forwarded-For`, de la préférence la plus élevée à la plus faible.

Vous pouvez définir des en-têtes personnalisés de type `X-Forwarded-For`. Le nom de l'en-tête HTTP qui contient l'adresse IP du client d'origine peut varier en fonction du fournisseur. Dans ce cas, le paramètre `xff_headers` permet d'introduire des en-têtes personnalisés dans l'inspecteur HTTP.

La valeur par défaut du paramètre `xff_headers` est `x-forwarded-for true-client-ip`, deux en-têtes connus. Si les deux en-têtes par défaut sont présents dans le flux, `x-forwarded-for` est préféré à `true-client-ip`. Lorsque vous spécifiez plusieurs en-têtes HTTP `X-Forwarded-For`, utilisez un espace pour délimiter leurs noms.

Type : chaîne

Valeurs valides : `x-forwarded-for`, `true-client-ip`

Valeur par défaut : `x-forwarded-for true-client-ip`

Règles de l'inspecteur HTTP Inspect

Activez les règles de l'inspecteur `http_inspect` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 14 : Règles de l'inspecteur HTTP Inspect

GID:SID	Message de règle
119:1	L'URI contient un caractère non réservé encodé en pourcentage
119:2	L'URI est encodé en pourcentage et le résultat est à nouveau encodé en pourcentage
119:3	L'URI contient un encodage Unicode de style %u non standard
119:4	L'URI contient des encodages Unicode avec des octets non encodés en pourcentage
119:6	L'URI contient un encodage UTF-8 sur deux ou trois octets
119:7	L'URI contient un encodage de point de code de la table Unicode
119:8	Le chemin de l'URI contient des caractères barre oblique consécutifs
119:9	Le caractère barre oblique inverse apparaît dans la partie chemin d'un URI
119:10	Le chemin de l'URI contient le modèle /./ répétant le répertoire actuel
119:11	Le chemin de l'URI contient le modèle /../ remontant d'un répertoire
119:12	Caractère tabulation dans la ligne de démarrage HTTP
119:13	Ligne de démarrage HTTP ou ligne d'en-tête se terminant par LF sans CR
119:14	L'URI normalisé comprend un caractère de la liste des caractères incorrects
119:15	Le chemin de l'URI contient un segment plus long que le paramètre oversize_dir_length
119:16	La longueur du bloc dépasse le maximum configuré dans l'option maximum_chunk_length
119:18	Le chemin de l'URI comprend /../ qui remonte au-delà du répertoire racine
119:19	La ligne d'en-tête HTTP dépasse 4 096 octets
119:20	Le message HTTP comporte plus de 200 champs d'en-tête
119:21	Le message HTTP comporte plusieurs valeurs pour l'en-tête Content-Length
119:24	Le champ d'en-tête Host apparaît plusieurs fois ou comporte plusieurs valeurs
119:25	La longueur de la valeur du champ d'en-tête HTTP Host dépasse l'option maximum_host_length

GID:SID	Message de règle
119:28	Demande HTTP POST ou PUT sans Content-Length ni blocs
119:31	La méthode de demande HTTP n'est pas connue de Snort
119:32	La demande HTTP utilise un format HTTP primitif appelé HTTP/0.9
119:33	L'URI de la demande HTTP comporte un espace non encodé en pourcentage
119:34	La connexion HTTP comporte plus de 100 demandes simultanées en pipeline qui n'ont pas reçu de réponse
119:102	Code d'état non valide dans la réponse HTTP
119:104	La réponse HTTP comporte un ensemble de caractères UTF qui n'a pas pu être normalisé
119:105	La réponse HTTP comporte un ensemble de caractères UTF-7
119:109	Multiples niveaux d'obscurcissement de JavaScript
119:110	Le nombre d'espaces JavaScript consécutifs dépasse le maximum autorisé
119:111	Encodages multiples dans les données JavaScript obscurcies
119:112	Échec de la décompression zlib du fichier SWF
119:113	Échec de la décompression LZMA du fichier SWF
119:114	Échec de la décompression deflate du fichier PDF
119:115	Fichier PDF avec un type de compression non pris en charge
119:116	Fichier PDF auquel plusieurs compressions sont appliquées
119:117	Échec de l'analyse du fichier PDF
119:201	Trafic autre que HTTP ou erreur de protocole HTTP irrécupérable
119:202	La longueur du bloc contient un nombre excessif de zéros
119:203	Espace avant ou entre les messages HTTP
119:204	Message de demande sans URI
119:205	Caractère de contrôle dans l'expression de motif de la réponse HTTP
119:206	Espace supplémentaire illégal dans la ligne de démarrage
119:207	Version HTTP corrompue
119:209	Erreur de format dans l'en-tête HTTP
119:210	Options d'en-tête de bloc présentes
119:211	URI mal formaté

GID:SID	Message de règle
119:212	Type non reconnu d'encodage en pourcentage dans l'URI
119:213	Bloc HTTP mal formaté
119:214	Espace adjacent à la longueur du bloc
119:215	Espace dans le nom de l'en-tête
119:216	Compression gzip excessive
119:217	Échec de la décompression gzip
119:218	Demande HTTP 0.9 suivie d'une autre demande
119:219	Demande HTTP 0.9 suivant une demande normale
119:220	Le message comporte à la fois Content-Length et Transfer-Encoding
119:221	Code d'état impliquant l'absence de corps combiné avec Transfer-Encoding ou Content-Length non nul
119:222	Transfer-Encoding ne se terminant pas par chunked
119:223	Transfer-Encoding avec des encodages avant chunked
119:224	Trafic HTTP mal formaté
119:225	Content-Encoding non pris en charge utilisé
119:226	Content-Encoding inconnu utilisé
119:227	Plusieurs Content-Encodings appliqués
119:228	Réponse du serveur avant la demande du client
119:229	Décompression PDF/SWF/ZIP de la réponse du serveur trop volumineuse
119:230	Caractère non imprimable dans le nom de l'en-tête du message HTTP
119:231	Valeur Content-Length incorrecte dans l'en-tête HTTP
119:232	Ligne d'en-tête HTTP encapsulée
119:233	Ligne d'en-tête HTTP se terminant par CR sans LF
119:234	Bloc terminé par un séparateur non standard
119:235	Longueur du bloc se terminant par LF sans CR
119:236	Plusieurs réponses avec le code d'état 100
119:237	Code d'état 100 ne répondant pas à un en-tête Expect
119:238	Code d'état 1XX autre que 100 ou 101

GID:SID	Message de règle
119:239	En-tête Expect envoyé sans corps de message
119:240	Message HTTP 1.0 avec en-tête Transfer-Encoding
119:241	Content-Transfer-Encoding utilisé comme en-tête HTTP
119:242	Champ illégal dans les trailers des messages segmentés
119:243	Champ d'en-tête apparaissant de manière inappropriée deux fois ou comportant deux valeurs
119:244	Valeur segmentée non valide dans l'en-tête Content-Encoding
119:245	Réponse 206 envoyée à une demande sans en-tête Range
119:246	HTTP dans le champ de version pas entièrement en majuscules
119:247	Espace intégré dans une valeur d'en-tête critique
119:248	Données compressées gzip suivies de données autres que gzip inattendues
119:249	Répétitions excessives de la clé de paramètre HTTP
119:253	Demande HTTP CONNECT avec un corps de message
119:254	Trafic HTTP client-serveur après une demande CONNECT, mais avant la réponse
119:255	Réponse HTTP CONNECT 2XX avec en-tête Content-Length
119:256	Réponse HTTP CONNECT 2XX avec en-tête Transfer-Encoding
119:257	Réponse HTTP CONNECT avec code d'état 1XX
119:258	Réponse HTTP CONNECT avant la fin du message de demande
119:259	Paramètre filename mal formé dans l'en-tête HTTP Content-Disposition
119:260	Le corps du message HTTP Content-Length a été tronqué
119:261	Le corps du message HTTP segmenté a été tronqué
119:262	Schéma d'URI HTTP de plus de 10 caractères
119:263	Un client HTTP/1 a demandé une mise à niveau vers HTTP/2
119:264	Un serveur HTTP/1 a accordé une mise à niveau vers HTTP/2
119:265	Jeton incorrect dans JavaScript
119:266	Balise d'ouverture de script inattendue dans le code JavaScript
119:267	Balise de fermeture de script inattendue dans le code JavaScript
119:268	Code JavaScript sous les balises de script externes

GID:SID	Message de règle
119:269	Balise d'ouverture de script sous une forme courte
119:270	Nombre maximal d'identifiants JavaScript uniques atteint
119:271	L'imbrication des crochets JavaScript dépasse la capacité
119:272	Virgules consécutives dans l'en-tête HTTP Accept-Encoding
119:273	PDU manquées lors de la normalisation de JavaScript
119:274	L'imbrication des portées JavaScript dépasse la capacité
119:275	Version HTTP/1 autre que 1.0 ou 1.1
119:276	La version de HTTP dans la ligne de démarrage est 0
119:277	La version de HTTP dans la ligne de démarrage est supérieure à 1

Options des règles de prévention des intrusions de l'inspecteur HTTP Inspect

http_client_body

Place le curseur de détection dans le corps d'une demande HTTP. Lorsqu'un message HTTP ne spécifie pas d'en-tête HTTP, Snort normalise `http_client_ DNS` à l'aide de la normalisation des URI. La normalisation des URI est généralement appliquée à `http_header`.

Syntaxe : `http_client_body;`

Exemples : `http_client_body;`

http_cookie

Place le curseur de détection sur le champ d'en-tête HTTP `Cookie` extrait. L'option de règle `http_cookie` comprend les paramètres suivants : `http_cookie.request`, `http_cookie.with_header`, `http_cookie.with_body` et `http_cookie.with_trailer`.

Syntaxe : http_cookie: <parameter>, <parameter>

Exemples : `http_cookie: request;`

http_cookie.request

Établit une correspondance avec le témoin HTTP trouvé dans le message de la demande HTTP. Utilisez le témoin de la demande HTTP pour examiner la réponse HTTP. Le paramètre `http_cookie.request` est facultatif.

Syntaxe : `http_cookie: request;`

Exemples : `http_cookie: request;`

http_cookie.with_header

Spécifie que la règle peut uniquement examiner les en-têtes des messages HTTP. Le paramètre `http_cookie.with_header` est facultatif.

Syntaxe : `http_cookie: with_header;`

Exemples : `http_cookie: with_header;`

http_cookie.with_body

Spécifie qu'une autre partie de la règle examine le corps du message HTTP, et non l'option de règle `http_cookie`. Le paramètre `http_cookie.with_body` est facultatif.

Syntaxe : `http_cookie: with_body;`

Exemples : `http_cookie: with_body;`

http_cookie.with_trailer

Spécifie qu'une autre partie de la règle examine les trailers des messages HTTP, et non l'option de règle `http_cookie`. Le paramètre `http_cookie.with_trailer` est facultatif.

Syntaxe : `http_cookie: with_trailer;`

Exemples : `http_cookie: with_trailer;`

http_header

Place le curseur de détection sur les en-têtes HTTP normalisés. Vous pouvez spécifier des noms d'en-tête individuels à l'aide de l'option `field`.

L'option de règle `http_header` comprend les paramètres suivants : `http_header.field`, `http_header.request`, `http_header.with_header`, `http_header.with_body` et `http_header.with_trailer`.

Syntaxe : `http_header: field <field_name>,<parameter>, <parameter>`

Exemples : `http_header: field Content-Type, with_trailer;`

http_header.field

Établit une correspondance entre le nom d'en-tête spécifié et les en-têtes HTTP normalisés. Le nom d'en-tête n'est pas sensible à la casse. Si vous ne spécifiez pas de nom d'en-tête, l'inspecteur HTTP examine tous les en-têtes à l'exception des en-têtes de témoins HTTP (`Cookie` et `Set-Cookie`).

Type : chaîne

Syntaxe : `http_header: field <field_name>;`

Valeurs valides : un nom d'en-tête HTTP.

Exemples : `http_header: field Content-Type;`

http_header.request

Établit une correspondance avec les en-têtes trouvés dans la demande HTTP. Utilisez les en-têtes de la demande HTTP pour examiner la réponse HTTP. Le paramètre `http_header.request` est facultatif.

Syntaxe : `http_header: request;`

Exemples : `http_header: request;`

http_header.with_header

Spécifie que la règle peut uniquement examiner les en-têtes des messages HTTP. Le paramètre `http_header.with_header` est facultatif.

Syntaxe : `http_header: with_header;`

Exemples : `http_header: with_header;`

http_header.with_body

Spécifie qu'une autre partie de la règle examine le corps du message HTTP, et non l'option de règle `http_header`. Le paramètre `http_header.with_body` est facultatif.

Syntaxe : `http_header: with_body;`

Exemples : `http_header: with_body;`

http_header.with_trailer

Spécifie qu'une autre partie de la règle examine les trailers des messages HTTP, et non l'option de règle `http_header`. Le paramètre `http_header.with_trailer` est facultatif.

Syntaxe : `http_header: with_trailer;`

Exemples : `http_header: with_trailer;`

http_method

Place le curseur de détection sur la méthode de la demande HTTP. Les valeurs des méthodes de demande HTTP les plus couramment utilisées sont `GET`, `POST`, `OPTIONS`, `HEAD`, `DELETE`, `PUT`, `TRACE` et `CONNECT`.

L'option de règle `http_method` comprend les paramètres suivants : `http_method.with_header`, `http_method.with_body` et `http_method.with_trailer`.

Syntaxe : `http_method: <parameter>, <parameter>;`

Exemples : `http_method; content:"GET";`

http_method.with_header

Spécifie que la règle peut uniquement examiner les en-têtes des messages HTTP. Le paramètre `http_method.with_header` est facultatif.

Syntaxe : `http_method: with_header;`

Exemples : `http_method: with_header;`

http_method.with_body

Spécifie qu'une autre partie de la règle examine le corps du message HTTP, et non l'option de règle `http_header`. Le paramètre `http_method.with_body` est facultatif.

Syntaxe : `http_method: with_body;`

Exemples : `http_method: with_body;`

http_method.with_trailer

Spécifie qu'une autre partie de la règle examine les trailers des messages HTTP, et non l'option de règle `http_header`. Le paramètre `http_method.with_trailer` est facultatif.

Syntaxe : `http_method: with_trailer;`

Exemples : `http_method: with_trailer;`

http_param

Place le curseur de détection sur la clé de paramètre HTTP spécifiée. La clé de paramètre HTTP peut apparaître dans le corps de la demande.

L'option de règle `http_param` comprend les paramètres suivants : `http_param.param` et `http_method.nocase`.

Syntaxe : `http_param: <parameter_key>, nocase;`

Exemples : `http_param: offset, nocase;`

http_param.param

Établit une correspondance avec le paramètre spécifié.

Type : chaîne

Syntaxe : `http_param: <http_parameter>;`

Valeurs valides : un paramètre de demande ou un champ du corps de la demande.

Exemples : `http_param: offset;`

http_param.nocase

Établit une correspondance avec le paramètre spécifié, mais ne tient pas compte de la casse. Le paramètre `http_param.nocase` est facultatif.

Syntaxe : `http_param: nocase;`

Exemples : `http_param: nocase;`

http_raw_body

Place le curseur de détection sur le corps non normalisé du message de la demande ou de la réponse.

Syntaxe : `http_raw_body;`

Exemples : `http_raw_body;`

http_raw_cookie

Place le curseur de détection sur l'en-tête HTTP `Cookie` non normalisé. L'option de règle `http_raw_cookie` comprend les paramètres suivants : `http_raw_cookie.request`, `http_raw_cookie.with_header`, `http_raw_cookie.with_body` et `http_raw_cookie.with_trailer`.

Syntaxe : `http_raw_cookie: <parameter>, <parameter>;`

Exemples : `http_raw_cookie: request;`

http_raw_cookie.request

Établit une correspondance avec le témoin trouvé dans la demande HTTP. Utilisez le témoin de la demande HTTP pour examiner le message de réponse. Le paramètre `http_raw_cookie.request` est facultatif.

Syntaxe : `http_raw_cookie: request;`

Exemples : `http_raw_cookie: request;`

http_raw_cookie.with_header

Spécifie que la règle peut uniquement examiner les en-têtes des messages HTTP. Le paramètre `http_raw_cookie.with_header` est facultatif.

Syntaxe : `http_raw_cookie: with_header;`

Exemples : `http_raw_cookie: with_header;`

http_raw_cookie.with_body

Spécifie qu'une autre partie de la règle examine le corps du message HTTP, et non l'option de règle `http_raw_cookie`. Le paramètre `http_raw_cookie.with_body` est facultatif.

Syntaxe : `http_raw_cookie: with_body;`

Exemples : `http_raw_cookie: with_body;`

http_raw_cookie.with_trailer

Spécifie qu'une autre partie de la règle examine les trailers des messages HTTP, et non l'option de règle `http_raw_cookie`. Le paramètre `http_raw_cookie.with_trailer` est facultatif.

Syntaxe : `http_raw_cookie: with_trailer;`

Exemples : `http_raw_cookie: with_trailer;`

http_raw_header

Place le curseur de détection sur les en-têtes non normalisés. `http_raw_header` comprend tous les noms et valeurs d'en-tête non modifiés dans le message d'origine.

L'option de règle `http_raw_header` comprend les paramètres suivants : `http_raw_header.field`, `http_raw_header.request`, `http_raw_header.with_header`, `http_raw_header.with_body` et `http_raw_header.with_trailer`.

Syntaxe : `http_raw_header: field <field_name>, <parameter>, <parameter>;`

Exemples : `http_raw_header: field Content-Type, with_trailer;`

http_raw_header.field

Établit une correspondance entre le nom d'en-tête spécifié et les en-têtes HTTP non normalisés. Le nom d'en-tête n'est pas sensible à la casse. Si vous ne spécifiez pas de nom d'en-tête, l'inspecteur HTTP examine tous les en-têtes à l'exception des en-têtes de témoins HTTP (Cookie et Set-Cookie).

Type : chaîne

Syntaxe : `http_raw_header: field <field_name>`

Valeurs valides : un nom d'en-tête HTTP.

Exemples : `http_raw_header: field Content-Type;`

http_raw_header.request

Établit une correspondance avec les en-têtes trouvés dans le message de la demande HTTP. Utilisez les en-têtes de la demande HTTP pour examiner le message de réponse. Le paramètre `http_raw_header.request` est facultatif.

Syntaxe : `http_raw_header: request;`

Exemples : `http_raw_header: request;`

http_raw_header.with_header

Spécifie que la règle peut uniquement examiner les en-têtes des messages HTTP. Le paramètre `http_raw_header.with_header` est facultatif.

Syntaxe : `http_raw_header: with_header;`

Exemples : `http_raw_header: with_header;`

http_raw_header.with_body

Spécifie qu'une autre partie de la règle examine le corps du message HTTP, et non l'option de règle `http_raw_header`. Le paramètre `http_raw_header.with_body` est facultatif.

Syntaxe : `http_raw_header: with_body;`

Exemples : `http_raw_header: with_body;`

http_raw_header.with_trailer

Spécifie qu'une autre partie de la règle examine les trailers des messages HTTP, et non l'option de règle `http_raw_header`. Le paramètre `http_raw_header.with_trailer` est facultatif.

Syntaxe : `http_raw_header: with_trailer;`

Exemples : `http_raw_header: with_trailer;`

http_raw_request

Place le curseur de détection sur la ligne de demande non normalisée. Pour examiner une partie donnée de la première ligne d'en-tête, utilisez l'une des options de règles suivantes : `http_method`, `http_raw_uri` ou `http_version`.

L'option de règle `http_raw_request` comprend les paramètres suivants : `http_raw_request.with_header`, `http_raw_request.with_body` et `http_raw_request.with_trailer`.

Syntaxe : `http_raw_request: <parameter>, <parameter>;`

Exemples : `http_raw_request: with_header;`

http_raw_request.with_header

Spécifie que la règle peut uniquement examiner les en-têtes des messages HTTP. Le paramètre `http_raw_request.with_header` est facultatif.

Syntaxe : `http_raw_request: with_header;`

Exemples : `http_raw_request: with_header;`

http_raw_request.with_body

Spécifie qu'une autre partie de la règle examine le corps du message HTTP, et non l'option de règle `http_raw_request`. Le paramètre `http_raw_request.with_body` est facultatif.

Syntaxe : `http_raw_request: with_body;`

Exemples : `http_raw_request: with_body;`

http_raw_request.with_trailer

Spécifie qu'une autre partie de la règle examine les trailers des messages HTTP, et non l'option de règle `http_raw_request`. Le paramètre `http_raw_request.with_trailer` est facultatif.

Syntaxe : `http_raw_request: with_trailer;`

Exemples : `http_raw_request: with_trailer;`

http_raw_status

Place le curseur de détection sur la ligne d'état non normalisée. Pour examiner une partie donnée de la ligne d'état, utilisez l'une des options de règles suivantes : `http_version`, `http_stat_code` ou `http_stat_msg`.

L'option de règle `http_raw_status` comprend les paramètres suivants : `http_raw_status.with_body` et `http_raw_status.with_trailer`.

Syntaxe : `http_raw_status: <parameter>, <parameter>;`

Exemples : `http_raw_status: with_body;`

http_raw_status.with_body

Spécifie qu'une autre partie de la règle examine le corps du message HTTP, et non l'option de règle `http_raw_status`. Le paramètre `http_raw_status.with_body` est facultatif.

Syntaxe : `http_raw_status: with_body;`

Exemples : `http_raw_status: with_body;`

http_raw_status.with_trailer

Spécifie qu'une autre partie de la règle examine les trailers des messages HTTP, et non l'option de règle `http_raw_status`. Le paramètre `http_raw_status.with_trailer` est facultatif.

Syntaxe : `http_raw_status: with_trailer;`

Exemples : `http_raw_status: with_trailer;`

http_raw_trailer

Place le curseur de détection sur les trailers HTTP non normalisés. Les trailers contiennent des informations sur le contenu du message. Ils ne sont pas disponibles lorsque la demande du client crée des en-têtes HTTP.

`http_raw_trailer` est identique à `http_raw_header`, sauf qu'il s'applique aux en-têtes de fin. Vous devez créer des règles distinctes pour inspecter les en-têtes et les trailers HTTP.

L'option de règle `http_raw_trailer` comprend les paramètres suivants : `http_raw_trailer.field`, `http_raw_trailer.request`, `http_raw_trailer.with_header`, `http_raw_trailer.with_body`.

Syntaxe : `http_raw_trailer: field <field_name>, <parameter>, <parameter>;`

Exemples : `http_raw_trailer: field <field_name>, request;`

http_raw_trailer.field

Établit une correspondance entre le nom de trailer spécifié et les trailers HTTP non normalisés. Le nom de trailer n'est pas sensible à la casse.

Type : chaîne

Syntaxe : `http_raw_trailer: field <field_name>;`

Valeurs valides : un nom de trailer HTTP.

Exemples : `http_raw_trailer: field trailer-timestamp;`

http_raw_trailer.request

Établit une correspondance avec les trailers trouvés dans le message de la demande HTTP. Utilisez les trailers de la demande HTTP pour examiner le message de réponse. Le paramètre `http_raw_trailer.request` est facultatif.

Syntaxe : `http_raw_trailer: request;`

Exemples : `http_raw_trailer: request;`

http_raw_trailer.with_header

Spécifie que la règle ne peut examiner que les en-têtes de réponse HTTP. Le paramètre `http_raw_trailer.with_header` est facultatif.

Syntaxe : `http_raw_trailer: with_header;`

Exemples : `http_raw_trailer: with_header;`

http_raw_trailer.with_body

Spécifie qu'une autre partie de la règle examine le corps du message de réponse HTTP, et non l'option de règle `http_raw_trailer`. Le paramètre `http_raw_trailer.with_body` est facultatif.

Syntaxe : `http_raw_trailer: with_body;`

Exemples : `http_raw_trailer: with_body;`

http_raw_uri

Place le curseur de détection sur l'URI non normalisé.

L'option de règle `http_raw_uri` comprend :

- `http_raw_uri.with_header`
- `http_raw_uri.with_body`
- `http_raw_uri.with_trailer`
- `http_raw_uri.scheme`

- `http_raw_uri.host`
- `http_raw_uri.port`
- `http_raw_uri.path`
- `http_raw_uri.query`
- `http_raw_uri.fragment`

Syntaxe : `http_raw_uri: <parameter>, <parameter>;`

Exemples : `http_raw_uri: with_header, path, query;`

http_raw_uri.with_header

Spécifie que la règle peut uniquement examiner les en-têtes des messages HTTP. Le paramètre `http_raw_uri.with_header` est facultatif.

Syntaxe : `http_raw_uri: with_header;`

Exemples : `http_raw_uri: with_header;`

http_raw_uri.with_body

Spécifie qu'une autre partie de la règle examine le corps du message HTTP, et non l'option de règle `http_raw_uri`. Le paramètre `http_raw_uri.with_body` est facultatif.

Syntaxe : `http_raw_uri: with_body;`

Exemples : `http_raw_uri: with_body;`

http_raw_uri.with_trailer

Spécifie qu'une autre partie de la règle examine les trailers des messages HTTP, et non l'option de règle `http_raw_uri`. Le paramètre `http_raw_uri.with_trailer` est facultatif.

Syntaxe : `http_raw_uri: with_trailer;`

Exemples : `http_raw_uri: with_trailer;`

http_raw_uri.scheme

N'établit une correspondance qu'avec le schéma de l'URI. Le paramètre `http_raw_uri.scheme` est facultatif.

Syntaxe : `http_raw_uri: scheme;`

Exemples : `http_raw_uri: scheme;`

http_raw_uri.host

N'établit une correspondance qu'avec l'hôte (nom de domaine) de l'URI. Le paramètre `http_raw_uri.host` est facultatif.

Syntaxe : `http_raw_uri: host;`

Exemples : `http_raw_uri: host;`

http_raw_uri.port

N'établit une correspondance qu'avec le port (TCP) de l'URI. Le paramètre `http_raw_uri.port` est facultatif.

Syntaxe : `http_raw_uri: port;`

Exemples : `http_raw_uri: port;`

http_raw_uri.path

N'établit une correspondance qu'avec la section chemin (répertoire et fichier) de l'URI. Le paramètre `http_raw_uri.path` est facultatif.

Syntaxe : `http_raw_uri: path;`

Exemples : `http_raw_uri: path;`

http_raw_uri.query

N'établit une correspondance qu'avec les paramètres de demande contenus dans l'URI. Le paramètre `http_raw_uri.query` est facultatif.

Syntaxe : `http_raw_uri: query;`

Exemples : `http_raw_uri: query;`

http_raw_uri.fragment

N'établit une correspondance qu'avec la section fragment de l'URI. Un fragment est une partie du fichier demandé, qui ne se trouve généralement que dans le navigateur et qui n'est pas transmise sur le réseau. Le paramètre `http_raw_uri.fragment` est facultatif.

Syntaxe : `http_raw_uri: fragment;`

Exemples : `http_raw_uri: fragment;`

http_stat_code

Place le curseur de détection sur le code d'état HTTP. Le code d'état HTTP est un nombre à trois chiffres compris entre 100 et 599.

L'option de règle `http_stat_code` comprend les paramètres suivants : `http_stat_code.with_body` et `http_stat_code.with_trailer`.

Syntaxe : `http_stat_code: <parameter>, <parameter>;`

Exemples : `http_stat_code: with_trailer;`

http_stat_code.with_body

Spécifie qu'une autre partie de la règle examine le corps du message HTTP, et non l'option de règle `http_stat_code`. Le paramètre `http_stat_code.with_body` est facultatif.

Syntaxe : `http_stat_code: with_body;`

Exemples : `http_stat_code: with_body;`

http_stat_code.with_trailer

Spécifie qu'une autre partie de la règle examine les trailers des messages HTTP, et non l'option de règle `http_stat_code`. Le paramètre `http_stat_code.with_trailer` est facultatif.

Syntaxe : `http_stat_code: with_trailer;`

Exemples : `http_stat_code: with_trailer;`

http_stat_msg

Place le curseur de détection sur le message d'état HTTP. Le message d'état HTTP décrit le code d'état HTTP en texte brut, par exemple : `OK`.

L'option de règle `http_stat_msg` comprend les paramètres suivants : `http_stat_msg.with_body` et `http_stat_msg.with_trailer`.

Syntaxe : `http_stat_msg: <parameter>, <parameter>;`

Exemples : `http_stat_msg: with_body;`

http_stat_msg.with_body

Spécifie qu'une autre partie de la règle examine le corps du message HTTP, et non l'option de règle `http_stat_msg`. Le paramètre `http_stat_msg.with_body` est facultatif.

Syntaxe : `http_stat_msg: with_body;`

Exemples : `http_stat_msg: with_body;`

http_stat_msg.with_trailer

Spécifie qu'une autre partie de la règle examine les trailers des messages HTTP, et non l'option de règle `http_stat_msg`. Le paramètre `http_stat_msg.with_trailer` est facultatif.

Syntaxe : `http_stat_msg: with_trailer;`

Exemples : `http_stat_msg: with_trailer;`

http_trailer

Place le curseur de détection sur les trailers normalisés. Les trailers contiennent des informations sur le contenu du message. Ils ne sont pas disponibles lorsque la demande du client crée des en-têtes HTTP.

`http_trailer` est identique à `http_header`, sauf qu'il s'applique aux en-têtes de fin. Vous devez créer des règles distinctes pour inspecter les en-têtes et les trailers HTTP.

L'option de règle `http_trailer` comprend les paramètres suivants : `http_trailer.field`, `http_trailer.request`, `http_trailer.with_header`, `http_trailer.with_body`.

Syntaxe : `http_trailer: field <field_name>, <parameter>, <parameter>;`

Exemples : `http_trailer: field trailer-timestamp, with_body;`

http_trailer.field

Établit une correspondance entre le nom de trailer spécifié et les trailers HTTP normalisés. Le nom de trailer n'est pas sensible à la casse.

Type : chaîne

Syntaxe : `http_trailer: field <field_name>;`

Valeurs valides : un nom de trailer HTTP.

Exemples : `http_trailer: field trailer-timestamp;`

http_trailer.request

Établit une correspondance avec les trailers trouvés dans le message de la demande HTTP. Utilisez les trailers de la demande HTTP pour examiner le message de réponse. Le paramètre `http_trailer.request` est facultatif.

Syntaxe : `http_trailer: request;`

Exemples : `http_trailer: request;`

http_trailer.with_header

Spécifie qu'une autre partie de la règle examine les en-têtes des messages HTTP, et non l'option de règle `http_trailer`. Le paramètre `http_trailer.with_header` est facultatif.

Syntaxe : `http_trailer: with_header;`

Exemples : `http_trailer: with_header;`

http_trailer.with_body

Spécifie qu'une autre partie de la règle examine le corps des messages HTTP, et non l'option de règle `http_trailer`. Le paramètre `http_trailer.with_body` est facultatif.

Syntaxe : `http_trailer: with_body;`

Exemples : `http_trailer: with_body;`

http_true_ip

Définissez le curseur de détection sur l'adresse IP du client final. Lorsqu'un client envoie une demande, le serveur proxy stocke l'adresse IP du client final. L'adresse IP d'un client est la dernière adresse IP répertoriée dans l'en-tête `X-Forwarded-For`, `True-Client-IP` ou tout autre type d'en-tête `X-Forwarded-For` personnalisé. En présence de plusieurs en-têtes, Snort prend en compte les en-têtes définis dans `xff_headers`.

L'option de règle `http_true_ip` comprend les paramètres suivants : `http_true_ip.with_header`, `http_true_ip.with_body` et `http_true_ip.with_trailer`.

Syntaxe : `http_true_ip: <parameter>, <parameter>;`

Exemples : `http_true_ip: with_header;`

http_true_ip.with_header

Spécifie que la règle peut uniquement examiner les en-têtes des messages HTTP. Le paramètre `http_true_ip.with_header` est facultatif.

Syntaxe : `http_true_ip: with_header;`

Exemples : `http_true_ip: with_header;`

http_true_ip.with_body

Spécifie qu'une autre partie de la règle examine le corps des messages HTTP, et non l'option de règle `http_true_ip`. Le paramètre `http_true_ip.with_body` est facultatif.

Syntaxe : `http_true_ip: with_body;`

Exemples : `http_true_ip: with_body;`

http_true_ip.with_trailer

Spécifie qu'une autre partie de la règle examine les trailers des messages HTTP, et non l'option de règle `http_true_ip`. Le paramètre `http_true_ip.with_trailer` est facultatif.

Syntaxe : `http_true_ip: with_trailer;`

Exemples : `http_true_ip: with_trailer;`

http_uri

Place le curseur de détection sur le tampon d'URI normalisé.

- `http_uri.with_header`
- `http_uri.with_body`
- `http_uri.with_trailer`
- `http_uri.scheme`
- `http_uri.host`
- `http_uri.port`
- `http_uri.path`
- `http_uri.query`
- `http_uri.fragment`

Syntaxe : `http_uri: <parameter>, <parameter>;`

Exemples : `http_uri: with_trailer, path, query;`

http_uri.with_header

Spécifie que la règle peut uniquement examiner les en-têtes des messages HTTP. Le paramètre `http_uri.with_header` est facultatif.

Syntaxe : `http_uri: with_header;`

Exemples : `http_uri: with_header;`

http_uri.with_body

Spécifie qu'une autre partie de la règle examine le corps du message HTTP, et non l'option de règle `http_uri`. Le paramètre `http_uri.with_body` est facultatif.

Syntaxe : `http_uri: with_body;`

Exemples : `http_uri: with_body;`

http_uri.with_trailer

Spécifie qu'une autre partie de la règle examine les trailers des messages HTTP, et non l'option de règle `http_uri`. Le paramètre `http_uri.with_trailer` est facultatif.

Syntaxe : `http_uri: with_trailer;`

Exemples : `http_uri: with_trailer;`

http_uri.scheme

N'établit une correspondance qu'avec le schéma de l'URI. Le paramètre `http_uri.scheme` est facultatif.

Syntaxe : `http_uri: scheme;`

Exemples : `http_uri: scheme;`

http_uri.host

N'établit une correspondance qu'avec l'hôte (nom de domaine) de l'URI. Le paramètre `http_uri.host` est facultatif.

Syntaxe : `http_uri: host;`

Exemples : `http_uri: host;`

http_uri.port

N'établit une correspondance qu'avec le port (TCP) de l'URI. Le paramètre `http_uri.port` est facultatif.

Syntaxe : `http_uri: port;`

Exemples : `http_uri: port;`

http_uri.path

N'établit une correspondance qu'avec le chemin (répertoire et fichier) de l'URI. Le paramètre `http_uri.path` est facultatif.

Syntaxe : `http_uri: path;`

Exemples : `http_uri: path;`

http_uri.query

N'établit une correspondance qu'avec les paramètres de demande contenus dans l'URI. Le paramètre `http_uri.query` est facultatif.

Syntaxe : `http_uri: uri;`

Exemples : `http_uri: query;`

http_uri.fragment

N'établit une correspondance qu'avec la section fragment de l'URI. Un fragment est une partie du fichier demandé, qui ne se trouve généralement que dans le navigateur et qui n'est pas transmise sur le réseau. Le paramètre `http_uri.fragment` est facultatif.

Syntaxe : `http_uri: fragment;`

Exemples : `http_uri: fragment;`

http_version

Place le curseur de détection au début du tampon de la version HTTP. `http_version` accepte différentes versions HTTP. Les versions les plus courantes sont : `HTTP/1.0` et `HTTP/1.1`. L'option de règle `http_version` comprend les paramètres suivants : `http_version.request`, `http_version.with_header`, `http_version.with_body` et `http_version.with_trailer`.

Syntaxe : `http_version: <parameter>, <parameter>;`

Exemples : `http_version; content:"HTTP/1.1";`

http_version.request

Établit une correspondance avec la version trouvée dans la demande HTTP. Utilisez la version de la demande pour examiner le message de réponse. Le paramètre `http_version.request` est facultatif.

Syntaxe : `http_version: request;`

Exemples : `http_version: request;`

http_version.with_header

Spécifie que la règle peut uniquement examiner les en-têtes des messages HTTP. Le paramètre `http_version.with_header` est facultatif.

Syntaxe : `http_version: with_header;`

Exemples : `http_version: with_header;`

http_version.with_body

Spécifie qu'une autre partie de la règle examine le corps du message HTTP, et non l'option de règle `http_version`. Le paramètre `http_version.with_body` est facultatif.

Syntaxe : `http_version: with_body;`

Exemples : `http_version: with_body;`

http_version.with_trailer

Spécifie qu'une autre partie de la règle examine les trailers des messages HTTP, et non l'option de règle `http_version`. Le paramètre `http_version.with_trailer` est facultatif.

Syntaxe : `http_version: with_trailer;`

Exemples : `http_version: with_trailer;`

http_version_match

Spécifie une liste de versions HTTP à comparer avec les versions HTTP standard. Utilisez un espace comme séparateur entre les différentes versions. Une demande HTTP ou une ligne d'état peut contenir une version. Si la version est présente, Snort la compare à la liste spécifiée dans `http_version_match`.

Si la version n'est pas au format `[0-9].[0-9]`, elle est considérée comme incorrecte. Toute version au format `[0-9].[0-9]` différente de `1.0` ou `1.1` est considérée comme `other`.

Type : chaîne

Syntaxe : `http_version_match: <version_list>`

Valeurs valides : `1.0, 1.1, 2.0, 0.9, other, malformed`

Exemples : `http_version_match: "1.0 1.1";`

js_data

Place le curseur de détection sur les données JavaScript normalisées. Cette option est propre au normalisateur JavaScript amélioré.

Syntaxe : `js_data;`

Exemples : `js_data;`

vba_data

Place le curseur de détection dans le tampon des macros de Microsoft Office Visual Basic for Applications.

Syntaxe : `vba_data;`

Exemples : `vba_data;`

CHAPITRE 12

Inspecteur IEC104

Présentation de l'inspecteur IEC104

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`stream_tcp`
Activé	`faux`

Le protocole IEC 60870-5-104 (IEC104) est une norme de communication destinée à l'échange de messages de télécommande entre les systèmes d'alimentation électrique. Le protocole IEC104 utilise le port TCP 2404.

L'inspecteur `iec104` détecte les messages IEC104 dans le trafic réseau. L'inspecteur `iec104` analyse et normalise les messages IEC104 en combinant un message réparti sur plusieurs trames ou en séparant plusieurs messages au sein d'une même trame.

Lorsqu'elles sont activées, les options de règles de prévention des intrusions permettent d'accéder au type d'informations de contrôle du protocole d'application (APCI) IEC104 et au code de fonction de l'unité de données de service d'application (ASDU).

Paramètres de l'inspecteur IEC104

Configuration du port TCP IEC104

L'inspecteur `binder` configure le port TCP IEC104. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
    {
        "when": {
            "role": "server",
            "proto": "tcp",
            "ports": "2404"
         },
        "use": {
            "type": "iec104"
        }
    }
]
```

Remarque L'inspecteur `IEC104` ne fournit aucun paramètre.

Règles de l'inspecteur IEC104

Activez les règles de l'inspecteur `iec104` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 15 : Règles de l'inspecteur IEC104

GID:SID	Message de règle
151:1	La longueur indiquée dans l'en-tête de l'APCI IEC104 ne correspond pas à la longueur nécessaire pour l'ID de type d'ASDU IEC104 donné
151:2	L'octet de début IEC104 ne correspond pas à 0x68
151:3	ID de type d'ASDU IEC104 réservé en cours d'utilisation
151:4	Le champ réservé de l'APCI U IEC104 contient une valeur autre que la valeur par défaut
151:5	Le type de message APCI U IEC104 a été défini sur une valeur non valide
151:6	Le champ réservé de l'APCI S IEC104 contient une valeur autre que la valeur par défaut
151:7	Le nombre d'éléments de l'APCI I IEC104 est défini sur zéro
151:8	Le bit SQ de l'APCI I IEC104 est activé sur une ASDU qui ne prend pas en charge cette fonctionnalité
151:9	Le nombre d'éléments de l'APCI I IEC104 est supérieur à un sur une ASDU qui ne prend pas en charge cette fonctionnalité
151:10	La cause d'initialisation de l'APCI I IEC104 est définie sur une valeur réservée
151:11	Le qualificateur de la commande d'interrogation de l'APCI I IEC104 est défini sur une valeur réservée

GID:SID	Message de règle
151:12	Le qualificateur du paramètre de la demande de commande d'interrogation de compteur de l'APCI I IEC104 est défini sur une valeur réservée
151:13	Le qualificateur du paramètre du type de paramètre des valeurs mesurées de l'APCI I IEC104 est défini sur une valeur réservée
151:14	Le qualificateur du paramètre de changement de paramètre local des valeurs mesurées de l'APCI I IEC104 est défini sur une valeur techniquement valide mais non utilisée
151:15	Le qualificateur de l'option de paramètre des valeurs mesurées de l'APCI I IEC104 est défini sur une valeur techniquement valide mais non utilisée
151:16	Le qualificateur d'activation de paramètre de l'APCI I IEC104 est défini sur une valeur réservée
151:17	Le qualificateur de commande de l'APCI I IEC104 est défini sur une valeur réservée
151:18	Le qualificateur de processus de réinitialisation de l'APCI I IEC104 est défini sur une valeur réservée
151:19	Le qualificateur de fichier prêt de l'APCI I IEC104 est défini sur une valeur réservée
151:20	Le qualificateur de section prête de l'APCI I IEC104 est défini sur une valeur réservée
151:21	Le qualificateur de sélection et d'appel de l'APCI I IEC104 est défini sur une valeur réservée
151:22	Le qualificateur de dernière section ou de dernier segment de l'APCI I IEC104 est défini sur une valeur réservée
151:23	Le qualificateur d'accusé de réception de fichier ou de section de l'APCI I IEC104 est défini sur une valeur réservée
151:24	Le qualificateur de structure de l'APCI I IEC104 est présent sur un message où il ne devrait avoir aucun effet
151:25	Le champ réservé des informations de point unique de l'APCI I IEC104 contient une valeur autre que la valeur par défaut
151:26	Le champ réservé des informations de point double de l'APCI I IEC104 contient une valeur autre que la valeur par défaut
151:27	La cause de transmission de l'APCI I IEC104 est définie sur une valeur réservée
151:28	La cause de transmission de l'APCI I IEC104 est définie sur une valeur non autorisée pour l'ASDU
151:29	Une valeur d'adresse commune de deux octets non valide a été détectée dans l'APCI I IEC104
151:30	Le champ réservé de la structure du descripteur de qualité de l'APCI I IEC104 contient une valeur autre que la valeur par défaut

GID:SID	Message de règle
151:31	Le champ réservé de la structure du descripteur de qualité pour les événements d'équipement de protection de l'APCI I IEC104 contient une valeur autre que la valeur par défaut
151:32	Une valeur IEEE STD 754 de l'APCI I IEC104 se traduit par NaN
151:33	Une valeur IEEE STD 754 de l'APCI I IEC104 se traduit par une valeur infinie
151:34	Le champ réservé de la structure d'événement unique d'équipement de protection de l'APCI I IEC104 contient une valeur autre que la valeur par défaut
151:35	Le champ réservé de la structure d'événement de démarrage d'équipement de protection de l'APCI I IEC104 contient une valeur autre que la valeur par défaut
151:36	Le champ réservé de la structure d'information de circuit de sortie de l'APCI I IEC104 contient une valeur autre que la valeur par défaut
151:37	Un schéma de bits de test fixe anormal a été détecté dans l'APCI I IEC104
151:38	Le champ réservé de la structure de commande simple de l'APCI I IEC104 contient une valeur autre que la valeur par défaut
151:39	La structure de commande double de l'APCI I IEC104 contient une valeur non valide
151:40	Le champ réservé de la structure de commande de réglage de pas de l'APCI I IEC104 contient une valeur autre que la valeur par défaut
151:41	La valeur de la milliseconde de Time2a de l'APCI I IEC104 est en dehors de la plage autorisée
151:42	La valeur de la minute de Time2a de l'APCI I IEC104 est en dehors de la plage autorisée
151:43	Le champ réservé de la minute de Time2a de l'APCI I IEC104 contient une valeur autre que la valeur par défaut
151:44	La valeur de l'heure de Time2a de l'APCI I IEC104 est en dehors de la plage autorisée
151:45	Le champ réservé de l'heure de Time2a de l'APCI I IEC104 contient une valeur autre que la valeur par défaut
151:46	La valeur du jour du mois de Time2a de l'APCI I IEC104 est en dehors de la plage autorisée
151:47	La valeur du mois de Time2a de l'APCI I IEC104 est en dehors de la plage autorisée
151:48	Le champ réservé du mois de Time2a de l'APCI I IEC104 contient une valeur autre que la valeur par défaut
151:49	La valeur de l'année de Time2a de l'APCI I IEC104 est en dehors de la plage autorisée
151:50	Le champ réservé de l'année de Time2a de l'APCI I IEC104 contient une valeur autre que la valeur par défaut

GID:SID	Message de règle
151:51	Une valeur nulle de longueur de segment a été détectée dans l'APCI I IEC104
151:52	Une valeur non valide de longueur de segment a été détectée dans l'APCI I IEC104
151:53	L'état de fichier de l'APCI I IEC104 est défini sur une valeur réservée
151:54	Le champ ql du qualificateur de la commande de point de contrôle de l'APCI I IEC104 est défini sur une valeur réservée

Options des règles de prévention des intrusions de l'inspecteur IEC104

iec104_apci_type

Vérifie que le message IEC104 correspond au type de contrôle d'informations du protocole d'application (APIC) IEC104 défini dans l'option.

L'option de règle de prévention des intrusions `iec104_apci_type` accepte une chaîne spécifiée à l'aide du nom complet du type APIC, ou de l'abréviation du type APIC, en majuscules ou en minuscules.

Type : chaîne

Syntaxe : `iec104_apci_type: <apic_type>;`

Exemples :

```
iec104_apci_type: unnumbered_control_function;
iec104_apci_type: S;
iec104_apci_type: I;
iec104_apci_type: i;
```

iec104_asdu_func

Vérifie que le message IEC104 correspond au code de fonction de l'unité de données de service d'application (ASDU) IEC104 défini dans l'option.

L'option de règle de prévention des intrusions `iec104_asdu_func` accepte une chaîne spécifiée à l'aide du code de fonction ASDU, en majuscules ou en minuscules.

Type : chaîne

Syntaxe : `iec104_asdu_func: <asdu_func>;`

Exemples :

```
iec104_asdu_func: M_SP_NA_1;
iec104_asdu_func: m_sp_na_1;
```

CHAPITRE 13

Inspecteur IMAP

Présentation de l'inspecteur IMAP

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`stream_tcp`
Activé	`vrai`

Le protocole IMAP (Internet Message Application Protocol) permet aux clients de messagerie de récupérer des messages à partir d'un serveur IMAP3 distant. Un serveur IMAP3 utilise le port TCP 143 pour les sessions non sécurisées ou le port TCP 993 pour IMAP sur SSL/TLS.

L'inspecteur `imap` détecte le trafic IMAP et analyse les commandes et les réponses IMAP.

L'inspecteur `imap` segmente les messages IMAP en sections de commande, d'en-tête et de corps, et procède à l'extraction et au décodage des pièces jointes MIME (Multipurpose Internet Mail Extensions). Les pièces jointes MIME peuvent inclure plusieurs pièces jointes et des pièces jointes volumineuses réparties sur plusieurs paquets.

L'inspecteur `imap` identifie le trafic IMAP et l'ajoute à la liste d'autorisation de Snort. Lorsqu'elles sont activées, les règles de prévention des intrusions génèrent des événements sur le trafic IMAP anormal.

Paramètres de l'inspecteur IMAP

Configuration du service IMAP

L'inspecteur `binder` configure le `service` IMAP. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
    {
        "when": {
            "service": "imap",
            "role": any
        },
        "use": {
            "type": "imap"
        }
    }
]
```

b_64_decode_depth

Spécifie le nombre maximal d'octets à extraire et à décoder de chaque pièce jointe de courriel MIME codée en Base64. Vous pouvez spécifier un entier inférieur à 65 535, ou 0 pour désactiver le décodage. Spécifiez -1 pour n'appliquer aucune limite au nombre d'octets à décoder.

Vous pouvez activer la règle 141:4 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés en cas d'échec du décodage (en raison d'un encodage incorrect ou de données corrompues).

Type : entier

Plage valide : de `-1` à `65 535`

Valeur par défaut : `-1`

bitenc_decode_depth

Spécifie le nombre maximal d'octets à extraire de chaque pièce jointe MIME non codée. Vous pouvez spécifier un entier inférieur à 65 535, ou 0 pour désactiver l'extraction de la pièce jointe MIME non codée. Spécifiez -1 pour n'appliquer aucune limite au nombre d'octets à extraire. Ces types de pièces jointes englobent les formats 7 bits, 8 bits, binaires, ainsi que divers types de contenu multipartite tels que le texte brut, les images JPEG et PNG, et les fichiers MP4.

Type : entier

Plage valide : de `-1` à `65 535`

Valeur par défaut : `-1`

decompress_pdf

Indique si les fichiers `application/pdf` (PDF) contenus dans les pièces jointes MIME doivent être décompressés.

Vous pouvez activer la règle 141:8 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

decompress_swf

Indique si les fichiers `application/vnd.adobe.flash-movie` (SWF) contenus dans les pièces jointes MIME doivent être décompressés.

Vous pouvez activer la règle 141:8 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : entier

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

decompress_vba

Indique si les fichiers de macros Microsoft Office Visual Basic for Applications contenus dans les pièces jointes MIME doivent être décompressés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

decompress_zip

Indique si les fichiers `application/zip` (ZIP) contenus dans les pièces jointes MIME doivent être décompressés.

Vous pouvez activer la règle 141:8 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

qp_decode_depth

Spécifie le nombre maximum d'octets à extraire et à décoder de chaque pièce jointe MIME de courriel codée en quoted-printable (QP). Vous pouvez spécifier un entier inférieur à 65 535, ou 0 pour désactiver le décodage. Spécifiez -1 pour n'appliquer aucune limite au nombre d'octets à décoder.

Vous pouvez activer la règle 141:5 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés en cas d'échec du décodage (en raison d'un encodage incorrect ou de données corrompues).

Type : entier

Plage valide : de `-1` à `65 535`

Valeur par défaut : `-1`

uu_decode_depth

Indique le nombre maximal d'octets à extraire et à décoder de chaque pièce jointe MIME encodée au format Unix-to-Unix (uuencode). Vous pouvez spécifier un entier inférieur à 65 535, ou 0 pour désactiver le décodage. Spécifiez -1 pour n'appliquer aucune limite au nombre d'octets à décoder.

Vous pouvez activer la règle 141:7 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés en cas d'échec du décodage (en raison d'un encodage incorrect ou de données corrompues).

Type : entier

Plage valide : de `-1` à `65 535`

Valeur par défaut : `-1`

Règles de l'inspecteur IMAP

Activez les règles de l'inspecteur `imap` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 16 : Règles de l'inspecteur IMAP

GID:SID	Message de règle
1411	Commande IMAP3 inconnue
141:2	Réponse IMAP3 inconnue
141:4	Échec du décodage en base64
141:5	Échec du décodage quoted-printable
141:7	Échec du décodage Unix-to-Unix
141:8	Échec de la décompression de fichier

Options des règles de prévention des intrusions de l'inspecteur IMAP

vba_data

Place le curseur de détection dans le tampon des macros de Microsoft Office Visual Basic for Applications.

Syntaxe : `vba_data;`

Exemples : `vba_data;`

CHAPITRE 14

Inspecteur MMS

Présentation de l'inspecteur MMS

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`stream_tcp`
Activé	`faux`

La norme IEC 61850 est une norme internationale qui définit les protocoles de communication des systèmes d'alimentation électrique. Le protocole MMS (Manufacturing Message Specification) est l'un des protocoles de la norme IEC 61850. MMS permet le transfert en temps réel de données SCADA (Supervisory Control and Data Acquisition) entre divers dispositifs de fabrication et de contrôle des processus. Le protocole MMS utilise le port TCP 102 pour échanger des messages entre clients et serveurs.

L'inspecteur `mms` détecte et analyse le trafic MMS. Un message MMS peut être composé de plusieurs unités de données de protocole (PDU) regroupées dans un seul paquet TCP, d'une seule PDU répartie sur plusieurs paquets TCP, ou d'une combinaison de ces deux configurations de messages. L'inspecteur `mms` normalise le trafic MMS pour présenter des messages MMS complets à un appareil.

Vous pouvez créer des règles Snort 3 pour les messages MMS sans avoir à décoder le protocole MMS. L'inspecteur `mms` analyse les couches OSI qui encapsulent le protocole MMS et permet d'accéder à certains champs du protocole MMS et au contenu des données par le biais d'options de règles. Pour plus de renseignements sur les options de règles MMS, consultez Options des règles de prévention des intrusions de l'inspecteur MMS, à la page 124

Paramètres de l'inspecteur MMS

Configuration du service MMS

L'inspecteur `binder` configure le `service` MMS. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
    {
        "when": {
            "service": "mms"
        },
        "use": {
            "type": "mms"
        }
    }
]
```

Règles de l'inspecteur MMS

Aucune règle n'est associée à l'inspecteur `mms`.

Options des règles de prévention des intrusions de l'inspecteur MMS

mms_data

Positionne le curseur de détection au début de l'unité de données de protocole (PDU) MMS, en contournant toutes les couches d'encapsulation OSI. Lorsqu'une règle de prévention des intrusions comprend `mms_data`, les options suivantes de la règle commencent à être traitées à partir de la PDU MMS.

Syntaxe : `mms_data;`

Exemples :

L'exemple suivant de règle de prévention des intrusions définit l'option de règle `mms_data`. L'option de règle `mms_data` positionne le curseur de détection au début de la PDU MMS et vérifie l'octet situé à cette position pour déterminer s'il correspond à la valeur d'un message `Initiate-Request`.

```
alert tcp ( \
msg: "PROTOCOL-SCADA MMS Initiate-Request"; \
flow: to_server, established; \
mms_data; \
content:"|A8|", depth 1; \
sid:1000000; \
)
```

mms_func

Compare le nom ou le numéro de la fonction fournie avec le champ `Confirmed Service` (service confirmé) dans la demande ou la réponse MMS. Une alerte est déclenchée lorsque le nom ou le numéro de la fonction MMS correspond au contenu du champ `Confirmed Service` (service confirmé).

Type : `chaîne`

Syntaxe : `mms_func <function>;`

Exemples :

L'exemple suivant de règle de prévention des intrusions définit l'option de règle `mms_func` et génère une alerte lorsque le service `Confirmed Service Request` (demande de service confirmé) correspond au nom de fonction fourni. En outre, `mms_func` active la fonctionnalité de recherche de modèle rapide pour identifier le message `Confirmed Service Request (0xA0)` (demande de service confirmé).

```
alert tcp ( \
msg: "PROTOCOL-SCADA MMS svc get_name_list"; \
flow: to_server, established; \
content:"|A0|"; \
mms_func: get_name_list; \
sid:1000000; \
)
```

L'exemple suivant de règle de prévention des intrusions définit l'option de règle `mms_func` et génère une alerte lorsque le message `GetNameList` correspond au numéro de la fonction.

```
alert tcp ( \
msg: "PROTOCOL-SCADA MMS svc get_name_list"; \
flow: to_server, established; \
content:"|A0|"; \
mms_func:1; \
sid:1000001; \
)
```

CHAPITRE 15

Inspecteur Modbus

Présentation de l'inspecteur Modbus

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`stream_tcp`
Activé	`faux`

Le protocole Modbus établit une norme de communication pour l'échange de messages entre un système SCADA (Supervisory Control and Data Acquisition) et un automate programmable industriel. Le protocole Modbus utilise le port TCP 502.

L'inspecteur `modbus` détecte et analyse les messages Modbus dans le trafic réseau. Lorsqu'elles sont activées, les options de règles de prévention des intrusions Modbus permettent d'accéder à certains champs du protocole Modbus.

Bonnes pratiques en matière de configuration de l'inspecteur Modbus

Si aucun appareil Modbus n'est activé sur votre réseau, vous ne devez pas activer l'inspecteur `modbus` dans une politique d'analyse de réseau que vous appliquez au trafic.

Paramètres de l'inspecteur Modbus

Configuration du port TCP Modbus

L'inspecteur `binder` configure le port TCP Modbus. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
    {
        "when": {
            "role": "server",
            "proto": "tcp",
            "ports": "502"
        },
        "use": {
            "type": "modbus"
        }
    },
    {
        "when": {
            "role": "any",
            "service:" "modbus"
        },
        "use": {
            "type":"modbus"
        }
    }
]
```

Remarque L'inspecteur `modbus` ne fournit aucun paramètre.

Règles de l'inspecteur Modbus

Activez les règles de l'inspecteur `modbus` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 17 : Règles de l'inspecteur Modbus

GID:SID	Message de règle
144:1	La longueur de l'en-tête MBAP Modbus ne correspond pas à la longueur requise pour la fonction donnée
144:2	L'ID du protocole Modbus est différent de zéro
144:3	Code de fonction Modbus réservé en cours d'utilisation

Options des règles de prévention des intrusions de l'inspecteur Modbus

Vous pouvez utiliser une option `modbus` seule ou en combinaison avec les options de règles de prévention des intrusions `content` et `byte_jump`.

modbus_data

Place le curseur de données au début du champ `Data` (données) de Modbus.

Syntaxe : `modbus_data;`

Exemples : `modbus_data;`

modbus_func

Vérifie que le champ `Function` (fonction) de Modbus contient le code de fonction Modbus spécifié. Vous pouvez définir un entier positif ou une chaîne littérale pour représenter un code de fonction Modbus.

Type : chaîne

Syntaxe : `modbus_func: <function>;`

Valeurs valides :

Tableau 18 : Valeurs des codes de fonction Modbus

Code	Chaîne
1	`read_coils`
2	`read_discrete_inputs`
3	`read_holding_registers`
4	`read_input_registers`
5	`write_single_coil`
6	`write_single_register`
7	`read_exception_status`
8	`diagnostics`
11	`get_comm_event_counter`
12	`get_comm_event_log`
15	`write_multiple_coils`
16	`write_multiple_registers`
17	`report_slave_id`

Code	Chaîne
20	read_file_record
21	write_file_record
22	mask_write_register
23	read_write_multiple_registers
24	read_fifo_queue
43	encapsulated_interface_transport

Exemples :

```
modbus_func: read_coils;
modbus_func: 8;
```

modbus_unit

Vérifie que l'ID d'unité Modbus du message correspond à l'ID d'unité spécifié. Vous pouvez définir un numéro pour représenter l'ID d'unité Modbus.

Type : entier

Syntaxe : `modbus_unit: <unit_id>;`

Plage valide : de `0` à `255`

Exemples :

```
modbus_unit: 1;
```

CHAPITRE 16

Inspecteur de normalisation

Présentation de l'inspecteur de normalisation

Type	Inspecteur (paquet)
Usage	Contexte
Type d'instance	Réseau
Autres inspecteurs requis	Aucun
Activé	`vrai`

L'inspecteur `normalizer` détecte et supprime les anomalies de protocole dans les paquets. L'inspecteur `normalizer` peut réduire au minimum les risques que des agresseurs créent des paquets pour échapper à la détection dans le cadre des déploiements en ligne.

Remarque Avant d'envoyer du trafic à partir de votre réseau, vous devez déployer les configurations appropriées sur les appareils gérés à l'aide d'interfaces routées, commutées ou transparentes, ou de paires d'interfaces en ligne.

Vous pouvez spécifier la normalisation de n'importe quelle combinaison de protocoles IPv4, IPv6, ICMPv4, ICMPv6 et TCP dans les paquets. L'inspecteur `normalizer` effectue les normalisations paquet par paquet et gère la plupart des normalisations. L'inspecteur `stream_tcp` gère les normalisations de paquets et de flux liées à l'état TCP, y compris la normalisation de la charge utile TCP.

La normalisation en ligne se déroule juste après le décodage et avant le traitement par d'autres inspecteurs. La normalisation se poursuit des couches de paquets internes vers les couches externes.

L'inspecteur `normalizer` ne génère pas d'événements. L'inspecteur `normalizer` prépare les paquets en vue de leur utilisation par d'autres inspecteurs et dans le cadre de déploiements en ligne. L'inspecteur permet

également de s'assurer que les paquets traités par le système sont les mêmes que ceux reçus par les hôtes de votre réseau.

Paramètres de l'inspecteur de normalisation

Localisez la portée de `normalizer` dans votre configuration pour définir les paramètres de l'inspecteur `normalizer`.

ip6

Efface l'indicateur `Reserved` (réservé) dans le trafic IPv6.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

icmp4

Efface l'indicateur `Reserved` (réservé) dans le trafic ICMPv4.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

icmp6

Efface l'indicateur `Reserved` (réservé) dans le trafic ICMPv6.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

ip4.base

Efface le sous-champ d'un seul bit `Reserved` (réservé) du champ d'en-tête IPv4 Flags (indicateurs IPv4) ainsi que le remplissage des paramètres. Résout les problèmes urgents de pointeur/indicateur. Nous vous recommandons d'activer `ip4.base`.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

ip4.df

Efface le sous-champ d'un seul bit `Don't Fragment` (ne pas fragmenter) du champ d'en-tête IPv4 Flags (indicateurs IPv4). Activez `ip4.df` pour permettre à un routeur en aval de fragmenter les paquets au lieu de les abandonner. Le paramètre `ip4.df` permet d'éviter les évasions qui créent des paquets à abandonner.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

ip4.rf

Effacez les bits `Reserved` (réservé) sur les paquets entrants.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

ip4.tos

Efface le champ d'un octet `Differentiated Services` (services différenciés), anciennement appelé `Type of Service` (type de service).

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

ip4.trim

Tronque les paquets avec une charge utile excédentaire à la longueur de datagramme spécifiée dans l'en-tête IP plus l'en-tête de couche 2 (par exemple, Ethernet), mais ne les tronque pas en dessous de la longueur de trame minimale.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp.base

Efface le sous-champ d'un seul bit `Reserved` (réservé) de l'en-tête TCP ainsi que les octets de remplissage des options. Résout les problèmes urgents de pointeur ou d'indicateur.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp.block

Indique si les paquets doivent être abandonnés lors de la normalisation TCP.

Lorsque ce paramètre est activé, Snort bloque les paquets TCP anormaux qui, s'ils étaient normalisés, seraient non valides et seraient probablement bloqués par l'hôte destinataire. Par exemple, Snort bloque tout paquet SYN transmis après une session établie.

Snort abandonne tout paquet qui correspond à l'une des règles de l'inspecteur de flux TCP suivantes, que ces règles soient activées ou non :

- 129:1

- 129:3
- 129:4
- 129:6
- 129:8
- 129:11
- 129:14 à 129:19

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp.ecn

Active la normalisation par paquet ou par flux des indicateurs `Explicit Congestion Notification (ECN).`

- Spécifiez `packet` pour effacer les indicateurs ECN paquet par paquet, quelle que soit la négociation.
- Spécifiez `stream` pour effacer les indicateurs ECN flux par flux si l'utilisation d'ECN n'a pas été négociée. Si vous spécifiez `stream`, vous devez activer `tcp.require_3whs` dans l'inspecteur de flux TCP pour que la normalisation ait lieu.
- Spécifiez `off` pour désactiver le paramètre `tcp.ecn`.

Type : énumération

Valeurs valides : `off`, `packet`, `stream`

Valeur par défaut : `off`

tcp.ips

Active la normalisation du champ de données TCP pour assurer la cohérence des données retransmises. Tout segment qui ne peut pas être réassemblé correctement est abandonné.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `true`

tcp.opts

Indique si des options TCP spécifiques que vous autorisez dans le trafic doivent être normalisées. Snort ne normalise pas les options que vous autorisez explicitement. Le système normalise les options que vous n'autorisez pas explicitement.

Snort autorise toujours les options TCP suivantes, car elles sont couramment utilisées pour optimiser les performances TCP :

- Taille de segment maximum (MSS)
- Échelle de la fenêtre

- Horodatage TCP

Snort n'autorise pas automatiquement d'autres options moins couramment utilisées.

Lorsque `tcp.opts` est activé, les normalisations du trafic TCP incluent les actions suivantes :

- Tous les octets d'option sont définis sur No Operation (aucune opération; option 1 de TCP), à l'exception de MSS, de la mise à l'échelle de fenêtre, de l'horodatage et de toutes les options explicitement autorisées.
- Les octets de l'horodatage sont définis sur No Operation (aucune opération) si l'horodatage est présent mais non valide, ou valide mais non négocié.
- Le paquet est bloqué si l'horodatage est négocié, mais absent.
- Le champ d'option de réponse Time Stamp Echo Reply (TSecr) (écho d'horodatage (TSecr)) est effacé si le bit de contrôle d'accusé de réception (ACK) n'est pas activé.
- Les options MSS et Window Scale (mise à l'échelle de fenêtre) sont définies sur No Operation (aucune opération; TCP option 1) si le bit de contrôle SYN n'est pas activé.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp.pad

Efface tous les octets de remplissage d'option.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp.req_pay

Efface le champ `Urgent Pointer` (pointeur urgent) de l'en-tête TCP et le bit de contrôle urgent (URG) en l'absence de charge utile.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp.req_urg

Efface le champ `Urgent Pointer` (pointeur urgent) de l'en-tête TCP 16 bits si le bit de contrôle urgent (URG) n'est pas défini.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp.req_urp

Efface le bit de contrôle `urgent (URG)` si le champ `Urgent Pointer` (pointeur urgent) de l'en-tête TCP n'est pas défini.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp.resv

Efface les bits `Reserved` (réservés) dans l'en-tête TCP.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp.trim_mss

Réduit le champ TCP `Data` (données) à la taille de segment maximum (MSS) si la charge utile est plus longue que MSS.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp.trim_rst

Efface les données du paquet RST.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp.trim_syn

Supprime les données des paquets de synchronisation TCP (SYN).

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp.trim_win

Réduit le champ TCP `Data` (données) à la taille spécifiée dans le champ `Window` (fenêtre).

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp.urp

Définit le champ `Urgent Pointer` (pointeur urgent) de l'en-tête TCP à deux octets sur la longueur de la charge utile si le pointeur est supérieur à la longueur de la charge utile.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

Règles de l'inspecteur de normalisation

Aucune règle n'est associée à l'inspecteur `normalizer`.

Options des règles de prévention des intrusions de l'inspecteur de normalisation

L'inspecteur `normalizer` ne comporte aucune option pour les règles de prévention des intrusions.

CHAPITRE 17

Inspecteur POP

Présentation de l'inspecteur POP

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`stream_tcp`
Activé	`vrai`

Le protocole POP3 (Post Office Protocol version 3) permet aux clients de messagerie de récupérer des messages sur un serveur POP3 distant. Un serveur POP3 utilise le port TCP 110 pour les sessions non sécurisées ou le port TCP 995 pour POP sur SSL/TLS.

L'inspecteur `pop` détecte le trafic POP, et analyse les commandes et les réponses POP.

L'inspecteur `pop` segmente les messages POP en sections de commande, d'en-tête et de corps, et procède à l'extraction et au décodage des pièces jointes MIME (Multipurpose Internet Mail Extensions). L'inspecteur `pop` traite les pièces jointes MIME, y compris les pièces jointes multiples et les pièces jointes volumineuses réparties sur plusieurs paquets.

L'inspecteur `pop` identifie les messages POP et les ajoute à la liste d'autorisation de Snort. Lorsqu'elles sont activées, les règles de prévention des intrusions génèrent des événements sur le trafic POP anormal.

Paramètres de l'inspecteur POP

Remarque Le décodage, ou l'extraction lorsque la pièce jointe MIME ne nécessite pas de décodage, peut inclure plusieurs pièces jointes et des pièces jointes volumineuses réparties sur plusieurs paquets.

La valeur la plus élevée est utilisée lorsque les valeurs des paramètres `b_64_decode_depth`, `bitenc_decode_depth`, `qp_decode_depth` ou `uu_decode_depth` sont différentes dans les politiques suivantes :

- La politique d'analyse du réseau par défaut
- Toute autre politique d'analyse de réseau personnalisée appelée par les règles d'analyse de réseau dans la même politique de contrôle d'accès

Configuration du service POP

L'inspecteur `binder` configure le `service` POP. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
    {
        "when": {
           "service": "pop",
           "role": any
        },
        "use": {
           "type": "pop"
        }
    }
]
```

b_64_decode_depth

Spécifie le nombre maximal d'octets à extraire et à décoder de chaque pièce jointe de courriel MIME codée en Base64. Vous pouvez spécifier un entier inférieur à 65 535, ou 0 pour désactiver le décodage. Spécifiez -1 pour n'appliquer aucune limite au nombre d'octets à décoder.

Vous pouvez activer la règle 142:4 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés en cas d'échec du décodage.

Type : entier

Plage valide : de `-1` à `65 535`

Valeur par défaut : `-1`

bitenc_decode_depth

Spécifie le nombre maximal d'octets à extraire de chaque pièce jointe MIME non codée. Vous pouvez spécifier un entier inférieur à 65 535, ou 0 pour désactiver l'extraction de la pièce jointe MIME non codée. Spécifiez -1 pour n'appliquer aucune limite au nombre d'octets à extraire. Ces types de pièces jointes englobent les formats 7 bits, 8 bits, binaires, ainsi que divers types de contenu multipartite tels que le texte brut, les images JPEG et PNG, et les fichiers MP4.

Type : entier

Plage valide : de `-1` à `65 535`

Valeur par défaut : `-1`

decompress_pdf

Indique si les fichiers `application/pdf` (PDF) contenus dans les pièces jointes MIME doivent être décompressés.

Vous pouvez activer la règle 142:8 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

decompress_swf

Indique si les fichiers `application/vnd.adobe.flash-movie` (SWF) contenus dans les pièces jointes MIME doivent être décompressés.

Vous pouvez activer la règle 142:8 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

decompress_vba

Indique si les fichiers de macros Microsoft Office Visual Basic for Applications contenus dans les pièces jointes MIME doivent être décompressés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

decompress_zip

Indique si les fichiers `application/zip` (ZIP) contenus dans les pièces jointes MIME doivent être décompressés.

Vous pouvez activer la règle 142:8 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

qp_decode_depth

Spécifie le nombre maximum d'octets à extraire et à décoder de chaque pièce jointe MIME de courriel codée en quoted-printable (QP). Vous pouvez spécifier un entier inférieur à 65 535, ou 0 pour désactiver le décodage. Spécifiez -1 pour n'appliquer aucune limite au nombre d'octets à décoder.

Vous pouvez activer la règle 142:5 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés en cas d'échec du décodage (en raison d'un encodage incorrect ou de données corrompues).

Type : entier

Plage valide : de `-1` à `65 535`

Valeur par défaut : `-1`

uu_decode_depth

Indique le nombre maximal d'octets à extraire et à décoder de chaque pièce jointe MIME encodée au format Unix-to-Unix (uuencode). Vous pouvez spécifier un entier inférieur à 65 535, ou 0 pour désactiver le décodage. Spécifiez -1 pour n'appliquer aucune limite au nombre d'octets à décoder.

Vous pouvez activer la règle 142:7 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés en cas d'échec du décodage (en raison d'un encodage incorrect ou de données corrompues).

Type : entier

Plage valide : de `-1` à `65 535`

Valeur par défaut : `-1`

Règles de l'inspecteur POP

Activez les règles de l'inspecteur `pop` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 19 : Règles de l'inspecteur POP

GID:SID	Message de règle
1421	Commande POP3 inconnue
142:2	Réponse POP3 inconnue
142:4	Échec du décodage en base64
142:5	Échec du décodage quoted-printable
142:7	Échec du décodage Unix-to-Unix
142:8	Échec de la décompression de fichier

Options des règles de prévention des intrusions de l'inspecteur POP

vba_data

Place le curseur de détection dans le tampon des macros de Microsoft Office Visual Basic for Applications.

Syntaxe : `vba_data;`

Exemples : `vba_data;`

CHAPITRE 18

Inspecteur d'analyse de ports

Présentation de l'inspecteur d'analyse de ports

Type	Inspecteur (sonde)
Usage	Global
Type d'instance	Global
Autres inspecteurs requis	Aucun
Activé	faux

Une analyse de ports est une méthode de reconnaissance du réseau souvent utilisée par les agresseurs comme prélude à une attaque. Lors d'une analyse de port, un agresseur envoie des paquets pour tester les protocoles et services réseau d'un hôte ciblé. En examinant les paquets envoyés en réponse par un hôte, l'agresseur peut déterminer quels ports sont ouverts sur celui-ci et, directement ou par déduction, quels protocoles d'application sont exécutés sur ces ports.

En soi, une analyse de ports n'est pas une preuve d'attaque. Les utilisateurs légitimes de votre réseau peuvent utiliser des techniques d'analyse de ports similaires à celles utilisées par les agresseurs.

L'inspecteur `port_scan` détecte quatre types d'analyse de ports et supervise les tentatives de connexion sur les protocoles TCP, UDP, ICMP et IP. En détectant des schémas d'activité, l'inspecteur `port_scan` vous aide à identifier les analyses de ports potentiellement malveillantes.

Tableau 20 : Types de protocoles d'analyse de ports

Protocole	Description
TCP	Détecte les sondes TCP telles que les analyses SYN, les analyses ACK, les analyses TCP connect() et les analyses utilisant des combinaisons d'indicateurs inhabituelles (comme Xmas tree, FIN et NULL).

Protocole	Description
UDP	Détecte les sondes UDP telles que les paquets UDP de zéro octet.
ICMP	Détecte les demandes ECHO ICMP (pings).
IP	Détecte les analyses de protocole IP. Au lieu de rechercher les ports ouverts, Snort recherche les protocoles IP pris en charge sur un hôte cible.

Les analyses de ports sont généralement divisées en quatre types, en fonction du nombre d'hôtes ciblés, du nombre d'hôtes à analyser et du nombre de ports qui sont analysés.

Tableau 21 : Types de balayage de ports

Type	Description
Balayage de ports	Il s'agit d'une analyse de ports de type un-à-un dans laquelle un agresseur utilise un ou plusieurs hôtes pour analyser plusieurs ports sur un seul hôte cible. Les analyses de ports de type un-à-un se caractérisent par : • un nombre réduit d'hôtes d'analyse • un hôte unique qui est analysé • un nombre élevé de ports analysés Une analyse de ports détecte les analyses de ports TCP, UDP et IP.
Balayage de ports multiples	Il s'agit d'un balayage de ports de type un-à-plusieurs dans lequel un agresseur utilise un ou plusieurs hôtes pour analyser un seul port sur plusieurs hôtes cibles. Les balayages de ports se caractérisent par : • un nombre réduit d'hôtes d'analyse • un nombre élevé d'hôtes analysés • un faible nombre de ports uniques analysés Un balayage de ports détecte les balayages de ports TCP, UDP, ICMP et IP.
Balayage de ports de leurre	Il s'agit d'une analyse de ports de type un-à-un dans laquelle l'agresseur associe de fausses adresses IP sources à l'adresse IP d'analyse réelle. Les analyses de ports avec leurres se caractérisent par : • un nombre élevé d'hôtes d'analyse • un faible nombre de ports qui ne sont analysés qu'une seule fois • un seul hôte analysé (ou un faible nombre) L'analyse de ports avec leurres détecte les analyses des ports des protocoles TCP, UDP et IP.

Type	Description
Balayage de ports distribués	Il s'agit d'une analyse de ports de type plusieurs-à-un dans laquelle plusieurs hôtes interrogent un seul hôte pour identifier les ports ouverts. Les analyses de ports distribuées se caractérisent par : • un nombre élevé d'hôtes d'analyse • un nombre élevé de ports qui ne sont analysés qu'une seule fois • un seul hôte analysé (ou un faible nombre) L'analyse de ports distribuée détecte les analyses de ports des protocoles TCP, UDP et IP.

Niveaux de sensibilité de l'analyse de ports

L'inspecteur `port_scan` propose trois niveaux de sensibilité par défaut :

- default_low_port_scan
- default_med_port_scan
- default_high_port_scan

Vous pouvez configurer des niveaux de sensibilité supplémentaires avec différents filtres :

- scans
- rejects
- nets
- ports

L'inspecteur `port_scan` identifie une sonde en analysant les réponses négatives des hôtes sondés. Par exemple, lorsqu'un client Web utilise TCP pour se connecter à un serveur Web, il peut supposer que le serveur Web écoute sur le port 80. Cependant, lorsqu'un agresseur sonde un serveur, il ne sait pas à l'avance si ce serveur offre des services Web. Lorsque l'inspecteur `port_scan` détecte une réponse négative (ICMP inaccessible ou paquet TCP RST), il enregistre cette réponse comme une potentielle analyse de ports. Le processus est plus difficile lorsque l'hôte ciblé se trouve de l'autre côté d'un périphérique tel qu'un pare-feu ou un routeur qui filtre les réponses négatives. Dans ce cas, l'inspecteur `port_scan` peut générer des événements d'analyse de ports filtrés en fonction du niveau de sensibilité que vous sélectionnez.

Bonnes pratiques en matière de configuration de l'inspecteur d'analyse de ports

Pour optimiser la détection des analyses de ports, nous vous recommandons de paramétrer l'inspecteur `port_scan` en fonction de la configuration de vos réseaux.

- Veillez à configurer correctement le paramètre `watch_ip`. Le paramètre `watch_ip` aide l'inspecteur `port_scan` à filtrer les hôtes légitimes qui sont très actifs sur votre réseau. Les adresses IP NAT, les serveurs de cache DNS, les serveurs syslog et les serveurs NFS en sont les exemples les plus courants.

- La plupart des faux-positifs que l'inspecteur `port_scan` peut générer sont de type `alert` d'analyse filtrée. Le type `alert` peut indiquer qu'un hôte a été trop actif pendant une période donnée. Si l'hôte génère continuellement le type `alert` d'analyse filtrée, ajoutez-le à la liste `ignore_scanners` ou réduisez le niveau de sensibilité de l'analyse.
- Utilisez le nombre de priorités, le nombre de connexions, le nombre d'adresses IP, le nombre de ports, la plage d'adresses IP et la plage de ports pour identifier les faux-positifs. Pour identifier les faux-positifs, la méthode la plus simple consiste à utiliser des estimations de rapports simples. La liste suivante présente des rapports à estimer et les valeurs qui leur sont associées pour différencier une analyse légitime d'un faux-positif.
  - Nombre de connexions/Nombre d'adresses IP : ce rapport fournit une estimation de la moyenne des connexions par adresse IP. Ce rapport doit être élevé pour les analyses de ports. Et il doit être faible pour les balayages de ports.
  - Nombre de ports/Nombre d'adresses IP : ce rapport fournit une estimation du nombre moyen de ports utilisés par adresse IP. Pour les analyses de ports, ce rapport doit être élevé, ce qui signifie que les ports de l'hôte analysé ont été utilisés par un nombre réduit d'adresses IP. Et pour les balayages de ports, il doit être faible, ce qui signifie que l'hôte analysé a utilisé peu de ports, mais sur un grand nombre d'hôtes.
  - Nombre de connexions/Nombre de ports : ce rapport fournit une estimation du nombre moyen de connexions par port. Ce rapport doit être faible pour les analyses de ports. Cela indique que chaque connexion s'est faite sur un port différent. Et il doit être élevé pour les balayages de ports. Cela indique que de nombreuses connexions ont été effectuées sur le même port.

  Plus le nombre de priorités est élevé, plus il est probable qu'il s'agisse d'une véritable analyse de ports ou d'un véritable balayage de ports (sauf si l'hôte est géré par un pare-feu).
- Si vous ne parvenez pas à détecter les analyses de ports, vous pouvez réduire le niveau de sensibilité de l'analyse. Le niveau de sensibilité le plus élevé offre la meilleure protection. Au niveau de sensibilité le plus bas, seules les réponses d'erreur déclenchent des alertes, et les analyses filtrées ne sont pas détectées. Les réponses d'erreur du niveau de sensibilité le plus bas peuvent signaler une analyse de ports, et les alertes générées sont très précises tout en nécessitant peu de réglages. Les analyses filtrées et les analyses du niveau de sensibilité élevé peuvent générer des faux-positifs.

Paramètres de l'inspecteur d'analyse de ports

memcap

Spécifie la mémoire maximale du suivi en octets.

Type : entier

Plage valide : de `1 024` à `9 007 199 254 740 992 (maxSZ)`

Valeur par défaut : `10 485 760`

protos

Spécifie les protocoles à superviser. Fournissez une chaîne d'abréviations de protocoles. Pour spécifier plusieurs protocoles, utilisez un espace comme séparateur entre les différentes abréviations.

Type : chaîne

Valeurs valides : `tcp`, `udp`, `icmp`, `ip`, `all`

Valeur par défaut : `all`

scan_types

Spécifie les types d'analyse de ports à examiner. Fournissez une chaîne d'abréviations de protocoles. Pour spécifier plusieurs protocoles, utilisez un espace comme séparateur entre les différentes chaînes.

Type : chaîne

Valeurs valides : `portscan`, `portsweep`, `decoy_portscan`, `distributed_portscan`, `all`

Valeur par défaut : `all`

watch_ip

Spécifie une liste de blocs CIDR et d'adresses IP avec des ports facultatifs à superviser.

Si `watch_ip` n'est pas défini, l'inspecteur `port_scan` examine tout le trafic réseau.

Type : chaîne

Valeurs valides : CIDR ou adresse IP, liste de CIDR ou d'adresses IP

Valeur par défaut : aucune

alert_all

Indique si une alerte doit être générée pour tous les événements de dépassement du seuil dans la fenêtre établie. Si `alert_all` est défini sur `false`, l'inspecteur `port_scan` ne génère une alerte qu'au premier événement de dépassement du seuil dans la fenêtre.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

include_midstream

Indique si une liste des CIDR avec ports facultatifs doit être créée.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

tcp_decoy.rejects

Spécifie le nombre de tentatives d'analyse avec réponses négatives.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `15`

tcp_decoy.ports

Spécifie le nombre de fois où le port (ou le protocole) a changé par rapport à une tentative précédente.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

tcp_decoy.scan

Spécifie le nombre de tentatives d'analyse.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `100`

tcp_decoy.nets

Spécifie le nombre de fois où l'adresse a changé par rapport aux tentatives précédentes.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

tcp_dist.rejects

Spécifie le nombre de tentatives d'analyse avec réponses négatives.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `15`

tcp_dist.ports

Spécifie le nombre de fois où le port (ou le protocole) a changé par rapport à une tentative précédente.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

tcp_dist.scans

Spécifie le nombre de tentatives d'analyse.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `100`

tcp_dist.nets

Spécifie le nombre de fois où l'adresse a changé par rapport aux tentatives précédentes.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

tcp_ports.rejects

Spécifie le nombre de tentatives d'analyse avec réponses négatives.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `15`

tcp_ports.ports

Spécifie le nombre de fois où le port (ou le protocole) a changé par rapport à une tentative précédente.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

tcp_ports.scans

Spécifie le nombre de tentatives d'analyse.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `100`

tcp_ports.nets

Spécifie le nombre de fois où l'adresse a changé par rapport aux tentatives précédentes.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

tcp_sweep.rejects

Spécifie le nombre de tentatives d'analyse avec réponses négatives.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `15`

tcp_sweep.ports

Spécifie le nombre de fois où le port (ou le protocole) a changé par rapport à une tentative précédente.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

tcp_sweep.scans

Spécifie le nombre de tentatives d'analyse.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `100`

tcp_sweep.nets

Spécifie le nombre de fois où l'adresse a changé par rapport aux tentatives précédentes.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

udp_decoy.rejects

Spécifie le nombre de tentatives d'analyse avec réponses négatives.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `15`

udp_decoy.ports

Spécifie le nombre de fois où le port (ou le protocole) a changé par rapport à une tentative précédente.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

udp_decoy.scans

Spécifie le nombre de tentatives d'analyse.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `100`

udp_decoy.nets

Spécifie le nombre de fois où l'adresse a changé par rapport aux tentatives précédentes.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

udp_dist.rejects

Spécifie le nombre de tentatives d'analyse avec réponses négatives.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `15`

udp_dist.ports

Spécifie le nombre de fois où le port (ou le protocole) a changé par rapport à une tentative précédente.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

udp_dist.scans

Spécifie le nombre de tentatives d'analyse.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `100`

udp_dist.nets

Spécifie le nombre de fois où l'adresse a changé par rapport aux tentatives précédentes.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

udp_ports.rejects

Spécifie le nombre de tentatives d'analyse avec réponses négatives.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `15`

udp_ports.ports

Spécifie le nombre de fois où le port (ou le protocole) a changé par rapport à une tentative précédente.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

udp_ports.scans

Spécifie le nombre de tentatives d'analyse.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `100`

udp_ports.nets

Spécifie le nombre de fois où l'adresse a changé par rapport aux tentatives précédentes.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

udp_sweep.rejects

Spécifie le nombre de tentatives d'analyse avec réponses négatives.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `15`

udp_sweep.ports

Spécifie le nombre de fois où le port (ou le protocole) a changé par rapport à une tentative précédente.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

udp_sweep.scans

Spécifie le nombre de tentatives d'analyse.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `100`

udp_sweep.nets

Spécifie le nombre de fois où l'adresse a changé par rapport aux tentatives précédentes.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

ip_decoy.rejects

Spécifie le nombre de tentatives d'analyse avec réponses négatives.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `15`

ip_decoy.ports

Spécifie le nombre de fois où le port (ou le protocole) a changé par rapport à une tentative précédente.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

ip_decoy.scans

Spécifie le nombre de tentatives d'analyse.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `100`

ip_decoy.nets

Spécifie le nombre de fois où l'adresse a changé par rapport aux tentatives précédentes.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

ip_dist.rejects

Spécifie le nombre de tentatives d'analyse avec réponses négatives.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `15`

ip_dist.ports

Spécifie le nombre de fois où le port (ou le protocole) a changé par rapport à une tentative précédente.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

ip_dist.scans

Spécifie le nombre de tentatives d'analyse.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `100`

ip_dist.nets

Spécifie le nombre de fois où l'adresse a changé par rapport aux tentatives précédentes.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

ip_sweep.rejects

Spécifie le nombre de tentatives d'analyse avec réponses négatives.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `15`

ip_sweep.ports

Spécifie le nombre de fois où le port (ou le protocole) a changé par rapport à une tentative précédente.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

ip_sweep.scans

Spécifie le nombre de tentatives d'analyse.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `100`

ip_sweep.nets

Spécifie le nombre de fois où l'adresse a changé par rapport aux tentatives précédentes.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

ip_proto.rejects

Spécifie le nombre de tentatives d'analyse avec réponses négatives.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `15`

ip_proto.ports

Spécifie le nombre de fois où le port (ou le protocole) a changé par rapport à une tentative précédente.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

ip_proto.scans

Spécifie le nombre de tentatives d'analyse.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `100`

ip_proto.nets

Spécifie le nombre de fois où l'adresse a changé par rapport aux tentatives précédentes.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

icmp_sweep.rejects

Spécifie le nombre de tentatives d'analyse avec réponses négatives.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `15`

icmp_sweep.ports

Spécifie le nombre de fois où le port (ou le protocole) a changé par rapport à une tentative précédente.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

icmp_sweep.scans

Spécifie le nombre de tentatives d'analyse.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `100`

icmp_sweep.nets

Spécifie le nombre de fois où l'adresse a changé par rapport aux tentatives précédentes.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `25`

tcp_window

Spécifie l'intervalle de détection pour les analyses du protocole TCP (Transmission Control Protocol).

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `0`

udp_window

Spécifie l'intervalle de détection pour les analyses du protocole UDP (User Datagram Protocol).

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `0`

ip_window

Spécifie l'intervalle de détection pour les analyses du protocole IP (Internet Protocol).

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `0`

icmp_window

Spécifie l'intervalle de détection pour les analyses du protocole ICMP (Internet Control Message Protocol).

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `0`

Règles de l'inspecteur d'analyse de ports

Activez les règles de l'inspecteur `port_scan` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 22 : Règles de l'inspecteur d'analyse de ports

GID:SID	Message de règle
122:1	Analyse de ports TCP
122:2	Analyse de ports TCP avec leurres
122:3	Balayage de ports TCP
122:4	Analyse de ports TCP distribuée
122:5	Analyse de ports TCP filtrée
122:6	Analyse de ports TCP avec leurres filtrée
122:7	Balayage de ports TCP filtré
122:8	Analyse de ports TCP distribuée et filtrée
122:9	Analyse de protocoles IP
122:10	Analyse de protocoles IP avec leurres
122:11	Balayage de protocoles IP
122:12	Analyse de protocoles IP distribuée
122:13	Analyse de protocoles IP filtrée
122:14	Analyse de protocoles IP avec leurres filtrée
122:15	Balayage de protocoles IP filtré
122:16	Analyse de protocoles IP distribuée et filtrée
122:17	Analyse de ports UDP
122:18	Analyse de ports UDP avec leurres
122:19	Balayage de ports UDP

GID:SID	Message de règle
122:20	Analyse de ports UDP distribuée
122:21	Balayage de ports UDP filtré
122:22	Analyse de ports UDP avec leurres filtrée
122:23	Balayage de ports UDP filtré
122:24	Analyse de ports UDP distribuée et filtrée
122:25	Balayage ICMP
122:26	Balayage ICMP filtré
122:27	Port ouvert

Options des règles de prévention des intrusions de l'inspecteur d'analyse de ports

L'inspecteur `port_scan` ne comporte aucune option pour les règles de prévention des intrusions.

CHAPITRE 19

Filtre de débit

Présentation du filtre de débit

Type	Module (de base)
Usage	Contexte
Type d'instance	Singleton
Activé	`faux`

Les attaques basées sur le débit tentent de submerger un réseau ou un hôte en lui envoyant un volume de trafic excessif, ce qui entraîne un ralentissement ou un refus des demandes légitimes. Vous pouvez utiliser la prévention basée sur le débit pour modifier l'action d'une règle de prévention des intrusions en réponse au nombre excessif de correspondances relatives à cette règle.

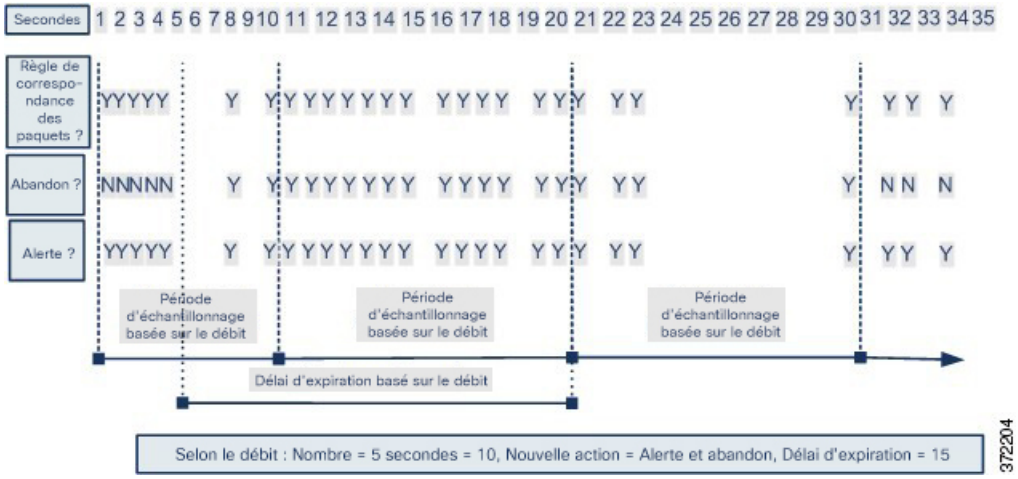
Le filtre `rate_filter` détecte les situations où une règle génère trop de correspondances pendant un intervalle donné. Vous pouvez utiliser cette fonctionnalité sur les périphériques gérés déployés en ligne pour bloquer les attaques basées sur le débit pendant une durée spécifiée, puis revenir à un état de règles où les correspondances de règles ne font que générer des événements et ne pas supprimer le trafic.

Vous pouvez configurer le filtre `rate_filter` pour qu'il répondre à n'importe quelle règle de prévention des intrusions, mais la règle que vous spécifiez doit être activée pour que `rate_filter` puisse détecter une attaque et y répondre. Par exemple, pour établir une défense contre une attaque DDOS/SYN flood, activez la règle 135:1 (TCP SYN reçu) et configurez le filtre `rate_filter` afin qu'il génère une alerte en cas de déclenchements excessifs de la règle 135:1.

La prévention des attaques basée sur le débit détecte les schémas de trafic anormaux et tente de minimiser l'impact de ce trafic sur les demandes légitimes. Vous pouvez repérer le nombre excessif de correspondances de règles dans le trafic dirigé vers une ou des adresses IP de destination en particulier ou provenant d'une ou d'adresses IP source en particulier. Vous pouvez également répondre au nombre excessif de correspondances pour une règle particulière dans tout le trafic détecté.

Le diagramme suivant montre un exemple dans lequel un agresseur tente d'accéder à un hôte. Les tentatives répétées pour trouver un mot de passe déclenchent une règle pour laquelle la prévention des attaques basée sur le débit est configurée. Les paramètres basés sur le débit remplacent l'attribut de règle par Abandon et génération d'événements après cinq correspondances de règles en 10 secondes. Le nouvel attribut de règle expire après 15 secondes.

Après l'expiration du délai, notez que les paquets sont toujours abandonnés durant la période d'échantillonnage basée sur le débit, qui suit. Si le débit échantillonné est supérieur au seuil au cours de la période d'échantillonnage en cours ou précédente, la nouvelle action se poursuit. La nouvelle action ne revient à Générer des événements qu'à la fin d'une période d'échantillonnage au cours de laquelle la fréquence échantillonnée était inférieure à la fréquence seuil.



Vous pouvez définir plusieurs filtres basés sur le débit sur la même règle ou sur des règles différentes. Dans une politique de prévention des intrusions comportant plusieurs filtres basés sur le débit, le premier filtre répertorié possède la priorité la plus élevée. En cas de conflit entre les actions de deux filtres basés sur le débit, l'action du premier filtre est exécutée.

Les paramètres de configuration définis pour le filtre `rate_filter` s'appliquent à l'ensemble du trafic de votre déploiement. Toutefois, pendant la période d'échantillonnage, le système compte séparément le nombre de correspondances de chaque connexion qu'il supervise. Le système gère également les modifications apportées à une action en les appliquant de façon distincte pour chaque connexion.

Remarque Les actions basées sur le débit ne peuvent pas activer les règles désactivées ni abandonner le trafic correspondant aux règles désactivées.

Paramètres du filtre de débit

rate_filter[]

Spécifie un tableau d'informations `rate_filter`. Chaque filtre de débit (`rate_filter`) comprend un ensemble de champs qui peuvent modifier l'action d'une règle en cas d'attaque basée sur le débit.

Type : tableau (objet)

Exemple :

```
{
    "rate_filter": {
        "data": [
            {
                "apply_to": "[10.1.2.100, 10.1.2.101]",
                "count": 5,
                "gid": 135,
                "new_action": "alert",
                "seconds": 1,
                "sid": 1,
                "timeout": 5,
                "track": "by_src"
            }
        ],
        "enabled": true,
        "type": "singleton"
    }
}
```

rate_filter[].gid

Spécifie un ID de générateur (GID) qui identifie la règle avec laquelle la correspondance doit être établie.

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `1`

rate_filter[].sid

Spécifie un ID de signature (SID) qui identifie la règle avec laquelle la correspondance doit être établie.

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `1`

rate_filter[].track

Spécifie un filtre pour les adresses source ou de destination.

Type : énumération

Valeurs valides :

- `by_src` : filtre uniquement le trafic correspondant à la règle spécifiée par `rate_filter[].gid` et `rate_filter[].sid`, et dont l'adresse source correspond à `rate_filter[].apply_to`.
- `by_dst` : filtre uniquement le trafic correspondant à la règle spécifiée par `gid` et `sid`, et dont l'adresse de destination correspond à `rate_filter[].apply_to`.
- `by_rule` : filtre tout le trafic correspondant à la règle spécifiée par `rate_filter[].gid` et `rate_filter[].sid`.

Valeur par défaut : `by_src`

rate_filter[].count

Spécifie le nombre de correspondances de règles à autoriser pendant la période d'échantillonnage (`rate_filter[].seconds`) avant d'appliquer l'action alternative (`rate_filter[].new_action`).

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `1`

rate_filter[].seconds

Spécifie le nombre de secondes de la période d'échantillonnage pour qu'elle corresponde au trafic. `rate_filter[].seconds` représente le délai qui doit s'écouler avant que le compteur interne de correspondances ne soit remis à zéro.

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `1`

rate_filter[].new_action

Spécifie l'action à exécuter en cas de détection dans le trafic de correspondances qui dépassent les limites définies par `rate_filter[].seconds` et `rate_filter[].count`.

Type : chaîne

Valeurs valides : l'une des chaînes suivantes : `alert`, `block`, `drop`, `log`, `pass`, `react`, `reject`, `rewrite`.

Valeur par défaut : `alert`

rate_filter[].timeout

Spécifie le nombre de secondes pendant lesquelles l'action indiquée par `rate_filter[].new_action` est exécutée en réponse au trafic correspondant.

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `0`

rate_filter[].apply_to

Spécifie la liste des adresses réseau utilisées pour la mise en correspondance avec l'adresse source ou de destination du trafic, en fonction de la valeur de `rate_filter[].track`.

Type : chaîne

Valeurs valides : une adresse IPv4 valide ou un bloc d'adresses IPv4 au format CIDR.

Valeur par défaut : aucune

Règles du filtre de débit

Aucune règle n'est associée au filtre `rate_filter`.

Vous pouvez configurer le filtre `rate_filter` pour qu'il réponde à toutes les règles de prévention des intrusions. Activez le filtre `rate_filter` afin qu'une règle détecte une attaque et y réponde.

Options des règles de prévention des intrusions du filtre de débit

Le filtre `rate_filter` ne comporte aucune option pour les règles de prévention des intrusions.

CHAPITRE 20

Inspecteur S7CommPlus

Présentation de l'inspecteur S7CommPlus

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`stream_tcp`
Activé	`faux`

S7CommPlus est un protocole propriétaire développé par Siemens. S7CommPlus permet la communication entre les automates programmables industriels de la gamme de produits Siemens S7.

L'inspecteur `s7commplus` détecte et analyse le trafic S7CommPlus. Vous pouvez définir des options de règles de prévention des intrusions pour déclencher des alertes basées sur les champs d'en-tête du code de fonction et d'opération S7CommPlus spécifiés, et ainsi détecter des attaques au sein du trafic S7CommPlus.

Bonnes pratiques en matière de configuration de l'inspecteur S7CommPlus

Si aucun appareil S7CommPlus n'est activé sur votre réseau, vous ne devez pas activer l'inspecteur `s7commplus` dans une politique d'analyse de réseau que vous appliquez au trafic.

Paramètres de l'inspecteur S7CommPlus

Configuration du port TCP S7CommPlus

L'inspecteur `binder` configure le port TCP S7CommPlus. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
    {
        "when": {
            "role": "server",
            "proto": "tcp",
            "ports": "102"
        },
        "use": {
            "type": "s7commplus"
        }
    },
    {
        "when": {
            "role": "any",
            "service": "s7commplus"
        },
        "use": {
            "type": "s7commplus"
        }
    }
]
```

Remarque L'inspecteur `S7CommPlus` ne fournit aucun paramètre.

Règles de l'inspecteur S7CommPlus

Activez les règles de l'inspecteur `s7commplus` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 23 : Règles de l'inspecteur S7CommPlus

GID:SID	Message de règle
149:1	La longueur de l'en-tête MBAP S7commplus ne correspond pas à la longueur requise pour la fonction S7commplus
149:2	L'ID du protocole S7commplus est différent de zéro
149:3	Code de fonction S7commplus réservé en cours d'utilisation

Options des règles de prévention des intrusions de l'inspecteur S7CommPlus

Vous pouvez utiliser les mots clés `s7commplus` seuls ou combinés pour créer des règles de prévention des intrusions personnalisées qui identifient les attaques ciblant le trafic détecté par l'inspecteur `s7commplus`. Pour les mots-clés configurables, spécifiez une valeur unique connue ou un seul entier dans la plage autorisée.

Tenez compte des points suivants :

- Lorsqu'une règle `s7commplus` comporte plusieurs mots-clés, une condition AND est appliquée entre ceux-ci.
- L'utilisation de plusieurs mots clés `s7commplus_func` ou `s7commplus_opcode` dans une même règle invalide cette règle. Une règle ainsi invalidée ne peut pas trouver de correspondance dans le trafic. Pour rechercher plusieurs valeurs avec ces mots clés, créez plusieurs règles.

s7commplus_content

Utilisez le mot clé `s7commplus_content` pour positionner le curseur de détection au début de la charge utile du paquet S7CommPlus. Nous vous recommandons de définir ce mot clé avant d'utiliser un mot clé `content` ou `protected_content` dans une règle de prévention des intrusions S7CommPlus.

Syntaxe : `s7commplus_content;`

Exemples : `s7commplus_content;`

s7commplus_func

Utilisez le mot clé `s7commplus_func` pour rechercher une correspondance avec l'un des paramètres d'en-tête S7CommPlus spécifiés. Vous pouvez spécifier le nom du paramètre S7CommPlus ou le code hexadécimal correspondant.

Type : chaîne

Syntaxe : `s7commplus_func: <header_parameter>;`

Valeurs valides :

Nom	Code
`explore`	`0x04BB`
`createobject`	`0x04CA`
`deleteobject`	`0x04D4`
`setvariable`	`0x04F2`
`getlink`	`0x0524`
`setmultivar`	`0x0542`
`getmultivar`	`0x054C`

Nom	Code
`beginsequence`	`0x0556`
`endsequence`	`0x0560`
`invoke`	`0x056B`
`getvarsubstr`	`0x0586`
`0x0` à `0xFF`	Notez que les expressions numériques permettent des valeurs supplémentaires.

Exemples : `s7commplus_func: createobject;`

s7commplus_opcode

Utilisez le mot clé `s7commplus_opcode` pour rechercher une correspondance avec l'un des paramètres d'en-tête S7CommPlus spécifiés. Vous pouvez spécifier le nom du paramètre S7CommPlus ou le code hexadécimal correspondant.

Type : chaîne

Syntaxe : `s7commplus_opcode: <header_parameter>`

Valeurs valides :

Nom	Code
`demande`	`0x31`
`response`	`0x32`
`notification`	`0x33`
`response2`	`0x02`
`0x0` à `0xFF`	Notez que les expressions numériques permettent des valeurs supplémentaires.

Exemples : `s7commplus_opcode: 0x31;`

CHAPITRE 21

Inspecteur SIP

Présentation de l'inspecteur SIP

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`stream_udp`
Activé	`vrai`

Le protocole SIP (Session Initiation Protocol) gère la création, la modification et la suppression des sessions d'appel en temps réel impliquant un ou plusieurs participants. SIP permet de contrôler des applications telles que la téléphonie sur Internet, les conférences multimédias, la messagerie instantanée, les jeux en ligne et le transfert de fichiers. Le protocole SIP est un protocole textuel de demande et de réponse.

Une demande SIP comprend un champ `method` qui identifie l'objet de la demande et un champ `Request-URI` qui indique où envoyer la demande. Un code d'état dans chaque réponse SIP indique le résultat de l'action demandée. Le protocole SIP utilise TCP (port 5060) ou UDP (port 5061).

Après avoir créé une session d'appel, SIP peut transmettre des flux audio et vidéo sur le protocole RTP (Real-Time Transport Protocol). Le corps du message SIP contient la négociation des paramètres du canal de données, l'annonce de la session et l'invitation à la session au format SDP (Session Description Protocol).

L'inspecteur `sip` détecte et analyse les messages SIP dans le trafic réseau. L'inspecteur `sip` extrait l'en-tête et le corps du message SIP, et transmet toutes les données contenues dans ce corps de message au moteur de détection.

L'inspecteur `sip` détecte les anomalies et les vulnérabilités connues au sein du trafic SIP, y compris les séquences d'appels désordonnées et non valides.

Remarque

- L'inspecteur `sip` ne décode pas les messages RTP. L'inspecteur `sip` identifie le canal RTP sur la base du port défini dans les données SDP.
- UDP achemine généralement les sessions multimédias prises en charge par SIP. L'inspecteur `sip` obtient les informations de suivi de session à partir du flux UDP décodé.
- Les options de règles SIP vous permettent de positionner le curseur de détection sur l'en-tête du paquet SIP ou sur le corps du message, et de limiter la détection aux paquets correspondant à des méthodes SIP ou à des codes d'état spécifiques.

Paramètres de l'inspecteur SIP

Configuration du service SIP

L'inspecteur `binder` configure le service SIP. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
    {
        "when": {
            "role": "any",
            "service": "sip"
        },
        "use": {
            "type": "sip"
        }
    }
]
```

ignore_call_channel

Indique si le trafic du canal de données audio/vidéo doit être inspecté. Lorsqu'il est activé, l'inspecteur `sip` décode l'ensemble du trafic du canal SIP hors données et ignore le trafic du canal SIP de données audio/vidéo.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

max_call_id_len

Spécifie le nombre maximal d'octets à autoriser dans le champ d'en-tête `Call-ID` (identifiant d'appel). Le champ `Call-ID` (identifiant d'appel) identifie de manière unique la session SIP dans les demandes et les réponses. L'inspecteur `sip` ne génère pas d'alerte lorsque la valeur de `max_call_id_len` est 0.

Vous pouvez activer la règle 140:5 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés. L'inspecteur `sip` génère un événement lorsque la longueur de l'en-tête `Call-ID` (identifiant d'appel) est supérieure à la valeur de `max_call_id_len`.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `256`

max_contact_len

Spécifie le nombre maximal d'octets à autoriser dans le champ d'en-tête `Contact`. Le champ `Contact` fournit un URI qui spécifie l'emplacement à contacter pour les messages suivants. L'inspecteur `sip` ne génère pas d'alerte lorsque la valeur est 0.

Vous pouvez activer la règle 140:15 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés. L'inspecteur `sip` génère un événement lorsque la longueur du champ d'en-tête `Contact` est supérieure à la valeur de `max_contact_len`.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `256`

max_content_len

Spécifie le nombre maximal d'octets à autoriser dans le contenu du corps du message. L'inspecteur `sip` ne génère pas d'alerte lorsque la valeur est 0.

Vous pouvez activer la règle 140:16 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés. L'inspecteur `sip` génère un événement lorsque la longueur du contenu est supérieure à la valeur de `max_content_len`.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `1 024`

max_dialogs

Spécifie le nombre maximal de boîtes de dialogue autorisé dans une session de flux. Si le nombre de boîtes de dialogue est supérieur à la limite fixée, l'inspecteur `sip` abandonne les plus anciennes jusqu'à ce qu'il ne dépasse plus le nombre maximal spécifié.

Vous pouvez activer la règle 140:27 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : entier

Plage valide : de `1` à `4 294 967 295 (max32)`

Valeur par défaut : `4`

max_from_len

Spécifie le nombre maximal d'octets à autoriser dans le champ d'en-tête `From` (de). Le champ `From` (de) identifie l'expéditeur du message. L'inspecteur `sip` ne génère pas d'alerte lorsque la valeur est 0.

Vous pouvez activer la règle 140:9 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés. L'inspecteur `sip` génère un événement lorsque la longueur du champ `From` (de) est supérieure à la valeur de `max_from_len`.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `256`

max_request_name_len

Spécifie le nombre maximal d'octets à autoriser dans le nom de la demande. Le nom de la demande SIP fait référence au nom de la méthode spécifiée dans l'identifiant de transaction SIP `CSeq`. L'inspecteur `sip` ne génère pas d'alerte lorsque la valeur est 0.

Vous pouvez activer la règle 140:7 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés. L'inspecteur `sip` génère un événement lorsque la longueur du nom de la demande est supérieure à la valeur de `max_request_name_len`.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `20`

max_requestName_len

Le paramètre `max_requestName_len` est obsolète. Remplacez-le par le paramètre `max_request_name_len`.

max_to_len

Spécifie le nombre maximal d'octets à autoriser dans le champ d'en-tête `To` (À). Le champ `To` (à) identifie le destinataire du message. L'inspecteur `sip` ne génère pas d'alerte lorsque la valeur est 0.

Vous pouvez activer la règle 140:11 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés. L'inspecteur `sip` génère un événement lorsque la longueur du champ `To` (à) est supérieure à la valeur de `max_to_len`.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `256`

max_uri_len

Spécifie le nombre maximal d'octets à autoriser dans le champ SIP `Request-URI` (URI de la demande). Le champ `Request-URI` (URI de la demande) indique le chemin d'accès à la ressource demandée. L'inspecteur `sip` ne génère pas d'alerte lorsque la valeur est 0.

Vous pouvez activer la règle 140:3 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés. L'inspecteur `sip` génère un événement lorsque la longueur du champ `Request-URI` (URI de la demande) est supérieure à la valeur de `max_uri_len`.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `256`

max_via_len

Spécifie le nombre maximal d'octets à autoriser dans le champ d'en-tête `Via`. Le champ `Via` identifie le mode de transport à utiliser dans la demande et l'emplacement du destinataire. L'inspecteur `sip` ne génère pas d'alerte lorsque la valeur est 0.

Vous pouvez activer la règle 140:13 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés. L'inspecteur `sip` génère un événement lorsque la longueur du champ `Via` est supérieure à la valeur de `max_via_len`.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `1 024`

methods

Spécifie une liste de méthodes SIP à détecter. Les noms des méthodes ne sont pas sensibles à la casse. Utilisez une virgule ou un espace comme séparateur entre les différents noms de méthodes de la liste. Un nom de méthode peut inclure des caractères alphabétiques, des chiffres et le caractère de soulignement.

Type : chaîne

Valeurs valides : `ack`, `benotify`, `bye`, `cancel`, `do`, `info`, `invite`, `join`, `message`, `notify`, `options`, `prack`, `publish`, `quath`, `refer`, `register`, `service`, `sprack`, `subscribe`, `unsubscribe`, `update`

Valeur par défaut : `invite cancel ack bye register options`

Règles de l'inspecteur SIP

Activez les règles de l'inspecteur `sip` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 24 : Règles de l'inspecteur SIP

GID:SID	Message de règle
140:2	URI de demande vide
140:3	URI trop long
140:4	Call-ID vide
140:5	Call-ID trop long
140:6	Numéro CSeq trop grand ou négatif
140:7	Nom de la demande trop long dans CSeq
140:8	En-tête From vide
140:9	En-tête From trop long
140:10	En-tête To vide

GID:SID	Message de règle
140:11	En-tête To trop long
140:12	En-tête Via vide
140:13	En-tête Via trop long
140:14	Contact vide
140:15	Contact trop long
140:16	Longueur du contenu trop grande ou négative
140:17	Plusieurs messages SIP dans un paquet
140:18	Incompatibilité de longueur de contenu
140:19	Nom de la demande non valide
140:20	Attaque par rejeu sur invitation
140:21	Modification illégale des informations de session
140:22	Le code d'état de la réponse n'est pas un nombre à 3 chiffres
140:23	En-tête Content-Type vide
140:24	Version de SIP non valide
140:25	Incompatibilité dans la METHOD de demande et l'en-tête CSEQ
140:26	Méthode inconnue
140:27	Nombre maximum de boîtes de dialogue atteint dans une session

Options des règles de prévention des intrusions de l'inspecteur SIP

sip_method

Une méthode de demande SIP identifie l'objet de la demande. Utilisez le mot clé `sip_method` pour trouver une correspondance de la méthode dans une demande SIP. Les noms des méthodes ne sont pas sensibles à la casse. Utilisez une virgule pour séparer les noms des méthodes.

Type : chaîne

Syntaxe : `sip_method: <methods>;`

Valeurs valides : `ack`, `benotify`, `bye`, `cancel`, `do`, `info`, `invite`, `join`, `message`, `notify`, `options`, `prack`, `publish`, `quath`, `refer`, `register`, `service`, `sprack`, `subscribe`, `unsubscribe`, `update`

Exemples : `sip_method: "ack,service,info,bye";`

sip_stat_code

Une réponse SIP comprend un code d'état à trois chiffres. Le code d'état SIP indique le résultat de l'action demandée. Utilisez le mot clé `sip_stat_code` pour trouver une correspondance entre une réponse SIP et les codes d'état spécifiés.

Vous pouvez spécifier un nombre à un chiffre qui représente le premier chiffre d'un code d'état à trois chiffres, un nombre à trois chiffres, ou une liste de nombres séparés par des virgules utilisant l'une ou l'autre combinaison de chiffres. Une liste correspond à un numéro de la liste qui correspond au code de la réponse SIP.

Type : entier

Syntaxe : `sip_stat_code: <codes>;`

Plages valides :

- de `1` à `9`
- de `100` à `999`

Exemples : `sip_stat_code: "1";`

Tableau 25 : Valeurs des paramètres SIP et codes d'état

Valeur du paramètre	Codes d'état détectés	Description
189	189	Définit un code d'état précis.
1	100 - 199	Définit un seul chiffre.
222, 3	222; 300 - 399	Définit une liste de nombres à trois chiffres ou à un chiffre, séparés par des virgules.

sip_header

Utilisez le mot clé `sip_header` pour positionner le curseur de détection au début du tampon d'en-tête SIP extrait. Limite l'inspection aux champs d'en-tête.

Syntaxe : `sip_header;`

Exemples : `sip_header;`

sip_body

Utilisez le mot clé `sip_body` pour positionner le curseur de détection au début du corps du message SIP extrait. Limite l'inspection au corps du message.

Syntaxe : `sip_body;`

Exemples : `sip_body;`

Remarque L'inspecteur `sip` extrait le corps entier du message et le met à la disposition du moteur de règles. Le moteur de règles ne se limite pas à la recherche de contenu SDP.

CHAPITRE 22

Inspecteur SMTP

Présentation de l'inspecteur SMTP

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`stream_tcp`
Activé	`vrai`

Le protocole SMTP (Simple Mail Transfer Protocol) permet à un client de messagerie d'envoyer des messages à un serveur de messagerie. SMTP émet des commandes pour transmettre un message à un destinataire. Un serveur SMTP utilise le port TCP 25 pour les sessions non sécurisées ou le port TCP 587 pour SMTP sur SSL/TLS.

L'inspecteur `smtp` détecte le trafic SMTP et analyse les commandes et les réponses SMTP.

L'inspecteur `smtp` segmente les messages SMTP en sections de commande, d'en-tête et de corps, et procède à l'extraction et au décodage des pièces jointes MIME (Multipurpose Internet Mail Extensions). Les pièces jointes MIME peuvent inclure plusieurs pièces jointes et des pièces jointes volumineuses réparties sur plusieurs paquets.

L'inspecteur `smtp` identifie les messages SMTP et les ajoute à la liste d'autorisation de Snort. Lorsqu'elles sont activées, les règles de prévention des intrusions génèrent des événements sur le trafic SMTP anormal.

Vous pouvez configurer l'inspecteur `smtp` de manière à :

- consigner les ID de courriel de l'expéditeur et du destinataire, les en-têtes de courriel et les noms des pièces jointes ainsi que tous les événements générés pour la session,

- normaliser les lignes de commande SMTP en supprimant les espaces superflus (l'inspecteur `smtp` normalise les espaces (ASCII 0x20) ou les tabulations (ASCII 0x09)),
- ignorer le trafic TLS chiffré pour améliorer les performances,
- ignorer les données de messagerie en texte brut pour améliorer les performances.

Bonnes pratiques en matière de configuration de l'inspecteur SMTP

Nous vous recommandons de suivre les directives de la RFC 2821 pour configurer les paramètres de base de l'inspecteur `smtp` :

- `max_command_line_len` : 512 caractères
- `max_header_line_len` : 1 024 caractères
- `max_response_line_len` : 512 caractères

Paramètres de l'inspecteur SMTP

Configuration du service SMTP

L'inspecteur `binder` configure le `service` SMTP. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
    {
        "when": {
            "service": "smtp",
            "role": any
        },
        "use": {
            "type": "smtp"
        }
    }
]
```

alt_max_command_line_len[]

Spécifie un tableau de commandes SMTP et une longueur de ligne maximale alternative pour chaque commande. La longueur de ligne maximale alternative remplace la valeur du paramètre `max_command_line_len` pour la commande SMTP. Vous pouvez activer la règle 124:4 pour afin de générer des événements pour ce paramètre.

Type : tableau

Exemple :

```
{
    "alt_max_command_line_len": [
        {
            "command": "AUTH",
```

```
            "length": 240
        }
    ]
}
```

alt_max_command_line_len[].command

Spécifie une chaîne de commande.

Type : chaîne

Valeurs valides : commande SMTP

Valeur par défaut : voir Tableau 26 : Commandes SMTP et longueurs alternatives par défaut des commandes.

alt_max_command_line_len[].length

Spécifie une longueur de ligne maximale alternative pour la commande. Spécifiez `0` pour désactiver la détection de la longueur de ligne d'une commande.

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : voir Tableau 26 : Commandes SMTP et longueurs alternatives par défaut des commandes.

Tableau 26 : Commandes SMTP et longueurs alternatives par défaut des commandes

Commande	Durée
ATRN	255
AUTH	246
BDAT	255
DONNÉES	246
DÉBOGAGE	255
EHLO	500
EMAL	255
ESAM	255
ESND	255
ESOM	255
ETRN	500
EVFY	255
EXPN	255
HELO	500
HELP	500

Commande	Durée
IDENT	255
MESSAGERIE	260
NOOP	255
ONEX	246
QUEU	246
QUIT	246
RCPT	300
RSET	255
SAML	246
ENVOYER	246
SIZE	255
SOML	246
STARTTLS	246
TICK	246
Webex	246
TURN	246
TURNME	246
VERB	246
VRFY	255
XADR	246
XAUTH	246
XCIR	246
XEXCH50	246
X-EXPS	246
XGEN	246
XLICENSE	246
X-LINK2STATE	246
XQUE	246

Commande	Durée
XSTA	246
XTRN	246
XUSR	246

auth_cmds

Spécifie une liste de commandes SMTP qui déclenchent l'échange d'authentification. Utilisez un espace comme séparateur entre les commandes SMTP.

Type : chaîne

Valeurs valides : commandes de déclenchement de l'échange d'authentification SMTP

Valeur par défaut : `AUTH XAUTH X-EXPS`

b64_decode_depth

Spécifie le nombre maximal d'octets à extraire et à décoder de chaque pièce jointe de courriel MIME codée en Base64. Vous pouvez spécifier un entier inférieur à 65 535, ou 0 pour désactiver le décodage. Spécifiez -1 pour n'appliquer aucune limite au nombre d'octets à décoder.

Vous pouvez activer la règle 124:10 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés en cas d'échec du décodage.

Type : entier

Plage valide : de `-1` à `65 535`

Valeur par défaut : `-1`

binary_data_cmds

Spécifie une liste de commandes SMTP qui déclenchent l'envoi de données et qui sont suivies d'une valeur de longueur (en octets) pour indiquer la quantité de données à envoyer. Utilisez un espace comme séparateur entre les commandes SMTP.

Type : chaîne

Valeurs valides : commandes SMTP valides de déclenchement d'envoi de données utilisant un argument de longueur de données

Valeur par défaut : `BDATA XEXCH50`

bitenc_decode_depth

Spécifie le nombre maximal d'octets à extraire de chaque pièce jointe MIME non codée. Vous pouvez spécifier un entier inférieur à 65 535, ou 0 pour désactiver l'extraction de la pièce jointe MIME non codée. Spécifiez -1 pour n'appliquer aucune limite au nombre d'octets à extraire. Ces types de pièces jointes englobent les formats 7 bits, 8 bits, binaires, ainsi que divers types de contenu multipartite tels que le texte brut, les images JPEG et PNG, et les fichiers MP4.

Type : entier

Plage valide : de `-1` à `65 535`

Valeur par défaut : `-1`

data_cmds

Spécifie une liste de commandes SMTP qui déclenchent l'envoi de données et utilisent un délimiteur de fin de données (<CRLF>.<CRLF>).

Type : chaîne

Valeurs valides : commande SMTP de déclenchement d'envoi de données utilisant un délimiteur de fin de données

Valeur par défaut : `DATA`

decompress_pdf

Indique si les fichiers `application/pdf` (PDF) contenus dans les pièces jointes MIME doivent être décompressés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

decompress_swf

Indique si les fichiers `application/vnd.adobe.flash-movie` (SWF) contenus dans les pièces jointes MIME doivent être décompressés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

decompress_vba

Indique si les fichiers de macros Microsoft Office Visual Basic for Applications contenus dans les pièces jointes MIME doivent être décompressés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

decompress_zip

Indique si les fichiers `application/zip` (ZIP) contenus dans les pièces jointes MIME doivent être décompressés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

email_hdrs_log_depth

Spécifie le nombre d'octets de l'en-tête de courriel à extraire des données SMTP. Spécifiez 0 pour désactiver l'extraction de l'en-tête de courriel.

Type : entier

Plage valide : de `0` à `20 480`

Valeur par défaut : `1 464`

ignore_data

Indique si la section des données de courriel doit être décodée (à l'exception des en-têtes MIME).

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

ignore_tls_data

Indique si les données chiffrées par TLS doivent être décodées.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

log_email_hdrs

Indique si l'en-tête de courriel SMTP et tous les événements générés pour la session doivent être décodés et enregistrés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

log_filename

Indique si les noms des fichiers joints MIME extraits de l'en-tête `Content-Disposition` dans le corps MIME doivent être décodés et enregistrés, de même que tous les événements générés pour la session. Si le message contient plusieurs pièces jointes MIME, l'inspecteur SMTP enregistre les noms de fichier en utilisant une virgule comme séparateur. L'inspecteur SMTP n'enregistre pas plus de 1 024 octets.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

log_mailfrom

Indique si l'adresse courriel de l'expéditeur extraite de la commande SMTP `MAIL FROM` et tous les événements générés pour la session doivent être décodés et enregistrés. Si le message contient plusieurs expéditeurs, l'inspecteur SMTP les enregistre en utilisant une virgule comme séparateur. L'inspecteur SMTP n'enregistre pas plus de 1 024 octets.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

log_rcptto

Indique si les adresses courriel des destinataires extraites de la commande SMTP `RCPT TO` et tous les événements générés pour la session doivent être décodés et enregistrés. Si le message contient plusieurs destinataires, l'inspecteur SMTP les enregistre en utilisant une virgule comme séparateur. L'inspecteur SMTP n'enregistre pas plus de 1 024 octets.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

max_auth_command_line_len

Spécifie le nombre maximal d'octets acceptés pour la ligne de commande d'authentification SMTP.

Vous pouvez activer la règle 124:15 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés. Spécifiez 0 pour désactiver les alertes relatives aux commandes SMTP `AUTH`, ou supprimez le paramètre `max_auth_command_line_len` de votre configuration Snort.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `1 000`

max_command_line_len

Spécifie le nombre maximal d'octets acceptés pour la ligne de commande SMTP.

La RFC 2821, qui est la spécification du groupe de travail en réseau relative au protocole SMTP, recommande une longueur maximale de 512 octets pour la ligne de commande. Spécifiez 0 pour désactiver les alertes relatives à la longueur de la ligne de commande SMTP, ou supprimez le paramètre `max_command_line_len` de votre configuration Snort.

Vous pouvez activer la règle 124:1 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `512`

max_header_line_len

Spécifie le nombre maximal d'octets acceptés pour la ligne d'en-tête des données SMTP.

La RFC 2821, qui est la spécification du groupe de travail en réseau relative au protocole SMTP, recommande une longueur maximale de 1 024 octets pour la ligne d'en-tête des données. Spécifiez 0 pour désactiver les alertes relatives à la longueur de la ligne d'en-tête des données SMTP, ou supprimez le paramètre `max_header_line_len` de votre configuration Snort.

Vous pouvez activer les règles 124:2 et 124:7 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `1 000`

max_response_line_len

Spécifie le nombre maximal d'octets acceptés pour la ligne de réponse SMTP.

La RFC 2821, qui est la spécification du groupe de travail en réseau relative au protocole SMTP, recommande une longueur maximale de 512 octets pour la ligne de réponse. Spécifiez 0 pour désactiver les alertes relatives à la longueur de la ligne de réponse SMTP, ou supprimez le paramètre `max_response_line_len` de votre configuration Snort.

Vous pouvez activer la règle 124:3 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `512`

normalize

Indique si la normalisation doit porter sur toutes les commandes, sur aucune commande ou sur une liste de commandes. Vous pouvez spécifier la liste des commandes dans le paramètre `normalize_cmds`. L'inspecteur vérifie qu'il n'y a pas plus d'un espace (ASCII 0x20) ou d'une tabulation (ASCII 0x09) après une commande.

Type : énumération

Valeurs valides :

- `none`
- `cmds`
- `all`

Valeur par défaut : `none`

normalize_cmds

Spécifie une liste de commandes SMTP à normaliser. Utilisez un espace comme séparateur entre les commandes SMTP.

Type : chaîne

Valeurs valides : commandes SMTP

Valeur par défaut : aucune

qp_decode_depth

Spécifie le nombre maximum d'octets à extraire et à décoder de chaque pièce jointe MIME de courriel codée en quoted-printable (QP). Vous pouvez spécifier un entier inférieur à 65 535, ou 0 pour désactiver le décodage. Spécifiez -1 pour n'appliquer aucune limite au nombre d'octets à décoder.

Vous pouvez activer la règle 124:11 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : entier

Plage valide : de `-1` à `65 535`

Valeur par défaut : `-1`

uu_decode_depth

Indique le nombre maximal d'octets à extraire et à décoder de chaque pièce jointe MIME encodée au format Unix-to-Unix (uuencode). Vous pouvez spécifier un entier inférieur à 65 535, ou 0 pour désactiver le décodage. Spécifiez -1 pour n'appliquer aucune limite au nombre d'octets à décoder.

Vous pouvez activer la règle 124:13 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés en cas d'échec du décodage (en raison d'un encodage incorrect ou de données corrompues, par exemple).

Type : entier

Plage valide : de `-1` à `65 535`

Valeur par défaut : `-1`

valid_cmds

Spécifie une liste supplémentaire de commandes SMTP que l'inspecteur SMTP considère comme valides.

L'inspecteur SMTP définit une liste de commandes SMTP valides par défaut : `ATRN AUTH BDAT DATA DEBUG EHLO EMAL ESAM ESND ESOM ETRN EVFY EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT RCPT RSET SAML SEND SIZE STARTTLS SOML TICK TIME TURN TURNME VERB VRFY X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR.`

Vous pouvez activer la règle 124:5 pour générer des événements et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : chaîne

Valeurs valides : commandes SMTP

Valeur par défaut : aucune

xlink2state

Indique comment l'inspecteur SMTP gère les paquets impliqués dans les attaques par dépassement de tampon X-Link2State de Microsoft Exchange (voir CVE-2005-0560 pour une description de la vulnérabilité). Vous pouvez désactiver la détection (`disable`), activer la détection et générer des alertes (`alert`), ou activer la détection et abandonner les paquets incriminés (`drop`).

Vous pouvez activer la règle 124:8 afin de générer des événements pour ce paramètre et, dans le cadre d'un déploiement en ligne, abandonner les paquets incriminés.

Type : énumération

Valeurs valides :

- `disable`
- `alert`
- `drop`

Valeur par défaut : `alert`

Règles de l'inspecteur SMTP

Activez les règles de l'inspecteur `smtp` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 27 : Règles de l'inspecteur SMTP

GID:SID	Message de règle
124:1	Tentative de dépassement de tampon de commande
124:2	Tentative de dépassement de tampon d'en-tête de données
124:3	Tentative de dépassement de tampon de réponse
124:4	Tentative de dépassement de tampon de commande spécifique
124:5	Commande inconnue
124:6	Commande illégale
124:7	Tentative de dépassement de tampon de nom d'en-tête
124:8	Tentative de dépassement de tampon de commande X-Link2State
124:10	Échec du décodage en base64
124:11	Échec du décodage quoted-printable
124:13	Échec du décodage Unix-to-Unix
124:14	Attaque d'authentification SASL de Cyberus
124:15	Tentative de dépassement de tampon de commande d'authentification
124:16	Échec de la décompression de fichier

Options des règles de prévention des intrusions de l'inspecteur SMTP

vba_data

Place le curseur de détection dans le tampon des macros de Microsoft Office Visual Basic for Applications.

Syntaxe : `vba_data;`

Exemples : `vba_data;`

CHAPITRE 23

SnortML

Type	Inspecteur (passif)
Usage	Inspecter
Type d'instance	Singleton
Autres inspecteurs requis	`snort_ml_engine, http_inspect`
Activé	Max Detect

Chaque jour, de nouvelles vulnérabilités sont découvertes dans des logiciels essentiels au fonctionnement du monde moderne. Les analystes en sécurité décortiquent ces nouvelles vulnérabilités, isolent les éléments nécessaires à leur déclenchement et créent des signatures pour détecter les exploits qui les ciblent. La plupart des signatures ne peuvent être conçues que pour des vulnérabilités spécifiques.

SnortML est un système de détection des exploits basé sur un réseau neuronal et conçu pour le système de prévention des intrusions Snort. Il est non seulement capable d'apprendre à détecter des attaques connues à partir de données d'entraînement, mais aussi d'apprendre à détecter des attaques qu'il n'a encore jamais rencontrées.

L'inspecteur `snort_ml` recherche principalement les attaques par injection SQL sur HTTP. Comme cet inspecteur peut avoir une incidence sur les performances, il n'est activé par défaut qu'en mode `Max Detect` (détection maximale).

- Règles SnortML, à la page 191
- Paramètres SnortML, à la page 192

Règles SnortML

Activez la règle de l'inspecteur `snort_ml` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés. La règle de l'inspecteur `snort_ml` n'est activée par défaut que dans le cadre de la politique NAP de détection maximale.

Tableau 28 : Règles de l'inspecteur Snort ML

GID:SID	Message de règle
411:1	(snort_ml) Menace potentielle identifiée dans les paramètres HTTP par un système de détection d'exploits basé sur les réseaux neuronaux

Paramètres SnortML

uri_depth

Spécifie le nombre d'octets à analyser dans l'URI HTTP. La valeur `-1` représente un nombre illimité.

Type : entier

Plage valide : de `-1` à `2 147 483 648`

Valeur par défaut : `-1`

client_body_depth

Spécifie le nombre d'octets à analyser dans le corps du client HTTP. La valeur `-1` représente un nombre illimité.

Type : entier

Plage valide : de `-1` à `2 147 483 648`

Valeur par défaut : `0`

CHAPITRE 24

Inspecteur SSH

Présentation de l'inspecteur SSH

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	Aucun
Activé	`vrai`

Le protocole SSH (Secure Shell) est un protocole réseau qui permet une communication sécurisée entre un client et un serveur sur un réseau non sécurisé. SSH prend en charge la tunnellisation et authentifie un hôte distant à l'aide de la cryptographie à clé publique.

Vous pouvez utiliser SSH pour transférer des fichiers en toute sécurité, ou vous connecter à un hôte distant et interagir avec la ligne de commande. Le protocole SSH utilise le port 22 sur TCP, UDP ou SCTP.

L'inspecteur `ssh` décode les paquets de flux et détecte les exploits SSH suivants :

- Exploit de type dépassement du tampon défi-réponse
- Exploit de type CRC32
- Exploit de type dépassement du client SSH SecureCRT
- Direction du message SSH incorrecte

Les attaques par dépassement du tampon défi-réponse et CRC-32 surviennent après l'authentification, lorsque la connexion réseau entre les hôtes est chiffrée. Les deux types d'attaques envoient une charge utile anormalement élevée de plus de 20 Ko au serveur immédiatement après la demande d'authentification.

L'inspecteur `ssh` détecte les attaques par dépassement du tampon défi-réponse et CRC-32 en comptant le nombre d'octets transmis au serveur. Si le nombre d'octets dépasse la limite définie dans un nombre prédéfini de paquets, l'inspecteur `ssh` génère une alerte. Les attaques CRC-32 s'appliquent uniquement à SSH version 1, tandis que les exploits de type dépassement du tampon défi-réponse s'appliquent uniquement à SSH version 2. L'inspecteur `ssh` lit la chaîne de version SSH au début de la session pour identifier le type d'attaque.

Les attaques par dépassement du tampon du client SSH SecureCRT et les attaques dues à une incompatibilité de protocole se produisent avant l'échange de clés, lorsque les hôtes tentent de sécuriser une connexion. L'attaque par dépassement du tampon du client SSH SecureCRT envoie une chaîne d'identifiant de protocole trop longue au client, ce qui provoque un dépassement de tampon. Une attaque due à une incompatibilité de protocole se produit lorsqu'une application cliente autre que SSH tente de se connecter à un serveur SSH sécurisé, ou lorsque les numéros de version du serveur et du client ne correspondent pas.

Remarque L'inspecteur `ssh` ne gère pas les attaques par force brute.

Bonnes pratiques en matière de configuration de l'inspecteur SSH

Nous vous recommandons d'utiliser les paramètres de configuration par défaut de l'inspecteur `ssh`. Si vous dépassez le nombre maximal de paquets chiffrés pour une session, tel que défini dans le paramètre `max_encrypted_packets`, l'inspecteur `ssh` cesse de traiter le trafic de cette session afin d'améliorer les performances. L'inspecteur `ssh` ne détecte que les vulnérabilités SSH qui apparaissent au début de la session SSH.

Remarque Si l'inspecteur `ssh` génère un faux-positif pour dépassement du tampon défi-réponse ou CRC 32, vous pouvez augmenter le nombre d'octets clients requis avec le paramètre `max_client_bytes`.

Paramètres de l'inspecteur SSH

Configuration du service SSH

L'inspecteur `binder` configure le `service` SSH. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
    {
        "when": {
            "service": "ssh",
            "role": any
        },
        "use": {
            "type": "ssh"
        }
```

```
    }
]
```

max_encrypted_packets

Spécifie le nombre maximal de paquets chiffrés à examiner avant que l'inspecteur `ssh` n'ignore une session SSH. Si vous dépassez le nombre maximal de paquets chiffrés pour une session, l'inspecteur `ssh` cesse de traiter le trafic de cette session afin d'améliorer les performances.

Type : entier

Plage valide : de `-1` à `65 535`

Valeur par défaut : `25`

max_client_bytes

Spécifie le nombre maximal d'octets sans réponse à transmettre au serveur avant que l'inspecteur `ssh` ne génère une alerte pour dépassement du tampon défi-réponse ou CRC 32. Si la limite `max_client_bytes` est dépassée avant l'envoi du nombre maximal de paquets chiffrés (`max_encrypted_packets`), l'inspecteur considère qu'une attaque a eu lieu et ignore le trafic.

Vous pouvez activer la règle 128:1 pour générer une alerte lorsque l'inspecteur détecte un dépassement du tampon défi-réponse ou la règle 128:2 pour générer une alerte lorsque l'inspecteur détecte un exploit CRC 32.

Pour chaque réponse valide que le client reçoit du serveur, l'inspecteur `ssh` réinitialise le nombre de paquets de `max_client bytes`.

Remarque Nous vous déconseillons de définir la valeur de `max_client_bytes` sur `0` ou `1`. Si vous définissez `max_client_bytes` sur `0` ou `1`, l'inspecteur `ssh` génère toujours une alerte.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : `19 600`

max_server_version_len

Spécifie la longueur maximale de la chaîne de version du serveur SSH. Si la longueur de la chaîne de version du serveur SSH dépasse la valeur de `max_server_version_len`, l'inspecteur `ssh` génère une alerte. Vous pouvez activer la règle 128:3 pour générer une alerte en cas de dépassement de la chaîne de version du serveur Secure CRT.

Type : entier

Plage valide : de `0` à `255`

Valeur par défaut : `80`

Remarque La configuration par défaut de l'inspecteur `ssh` n'active aucune alerte.

Règles de l'inspecteur SSH

Activez les règles de l'inspecteur `ssh` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 29 : Règles de l'inspecteur SSH

GID:SID	Message de règle
128:1	Exploit de type dépassement du tampon défi-réponse
128:2	Exploit de type SSH1 CRC32
128:3	Dépassement de la chaîne de version du serveur
128:5	Direction des messages incorrecte
128:6	Taille de la charge utile incorrecte pour la charge utile donnée
128:7	Échec de la détection de la chaîne de version SSH

Options des règles de prévention des intrusions de l'inspecteur SSH

L'inspecteur `ssh` ne comporte aucune option pour les règles de prévention des intrusions.

CHAPITRE 25

Inspecteur de flux ICMP

Présentation de l'inspecteur de flux ICMP

Type	Inspecteur (flux)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	Aucun
Activé	`vrai`

Le protocole ICMP (Internet Control Message Protocol) est un protocole de couche réseau utilisé par les applications utilitaires réseau et les appareils réseau. ICMP envoie des informations de diagnostic et d'erreur pour déterminer si la communication entre les hôtes IP a été établie ou si elle a échoué. Un message ICMP comprend un en-tête et des données.

Le protocole ICMP transmet des informations relatives à d'autres flux. Il ne transporte pas de données nécessitant un réassemblage et ne requiert pas de liaison basée sur la cible.

L'inspecteur `stream_icmp` définit le suivi des flux ICMP. Pour les messages Ping, l'inspecteur fournit un suivi de flux de base par le biais des champs d'adresse IP source et de destination et des champs de port de l'en-tête ICMP. Pour les destinations inaccessibles, l'inspecteur analyse les adresses IP d'origine et les ports de transport, puis il met à jour l'état de la session. L'inspecteur `port_scan` peut utiliser l'hôte et le port inaccessibles, s'ils sont disponibles.

Bonnes pratiques en matière de configuration de l'inspecteur de flux ICMP

Tenez compte des bonnes pratiques suivantes lors de la configuration de l'inspecteur `stream_icmp` :

- Créez un inspecteur `stream_icmp` pour chaque délai d'expiration de session que vous souhaitez appliquer à un hôte ou à un réseau. L'inspecteur `stream_icmp` associe le paramètre `session_timeout` aux réseaux ou aux hôtes ICMP définis dans l'inspecteur `binder`.

  Une même politique d'analyse de réseau peut contenir plusieurs versions de l'inspecteur `stream_icmp`.

Paramètres de l'inspecteur de flux ICMP

session_timeout

Spécifie le nombre de secondes pendant lesquelles l'inspecteur `stream_icmp` conserve un flux ICMP inactif dans la table d'état. Dès que Snort détecte un datagramme ICMP doté de la même clé de flux, il vérifie si la session du flux précédent a expiré. Si la session a expiré, Snort ferme le flux et en démarre un nouveau. Snort recherche les flux obsolètes associés à la configuration de flux de base.

Type : entier

Plage valide : de `0` à `2 147 483 647 (max31)`

Valeur par défaut : `60`

Règles de l'inspecteur de flux ICMP

Aucune règle n'est associée à l'inspecteur `stream_icmp`.

Options des règles de prévention des intrusions de l'inspecteur de flux ICMP

L'inspecteur `stream_icmp` ne comporte aucune option pour les règles de prévention des intrusions.

CHAPITRE 26

Inspecteur de flux IP

Présentation de l'inspecteur de flux IP

Type	Inspecteur (flux)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	Aucun
Activé	`vrai`

Le protocole IP (Internet Protocol) est un protocole de couche réseau sans connexion qui constitue la base d'Internet. Le protocole IP utilise des adresses d'hôte pour acheminer les messages d'un hôte source vers un hôte de destination sur les réseaux IP. L'adresse IP peut acheminer des paquets de données TCP et UDP, entre autres protocoles de transport.

Un message IP contient un en-tête et des données. L'en-tête IP comprend les adresses IP utilisées pour acheminer le message jusqu'à sa destination. La section des données IP encapsule la charge utile du message. Le protocole IP gère le réassemblage et la fragmentation des messages.

L'inspecteur `stream_ip` détecte un flux de réseau IP et examine les paquets de ce flux. L'inspecteur `stream_ip` définit les paramètres de configuration des sessions IP et du suivi des flux, de la politique du système d'exploitation et des chevauchements de datagrammes. Selon le mode, l'inspecteur `stream_ip` ou le plan de données Snort gère la défragmentation.

Bonnes pratiques en matière de configuration de l'inspecteur de flux IP

Tenez compte des bonnes pratiques suivantes lors de la configuration de l'inspecteur `stream_ip` :

- Créez un inspecteur `stream_ip` pour chaque configuration IP que vous souhaitez appliquer à un hôte, à un terminal ou à un réseau. L'inspecteur de flux IP associe la configuration IP aux hôtes, terminaux ou réseaux IP définis dans l'inspecteur `binder`.

  Une même politique d'analyse de réseau peut contenir plusieurs versions de l'inspecteur `stream_ip`.

Paramètres de l'inspecteur de flux IP

max_overlaps

Spécifie le nombre maximal de chevauchements autorisés pour chaque datagramme. Spécifiez `0` pour autoriser un nombre illimité de chevauchements.

Vous pouvez activer la règle 123:12 pour déclencher une alerte en cas de chevauchement excessif de fragments.

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : 0

min_frag_length

Spécifie le nombre minimal d'octets attendus dans le fragment IP. Spécifiez `0` pour autoriser un nombre illimité d'octets dans le fragment IP.

Vous pouvez activer la règle 123:13 afin de déclencher une alerte pour les fragments dont la longueur est inférieure à `min_frag_length`.

Type : entier

Plage valide : de `0` à `65 535`

Valeur par défaut : 0

min_ttl

Spécifie un nombre minimal de sauts ou une durée de vie (TTL) minimale. Supprimez les fragments dont la TTL est inférieure au minimum spécifié.

Vous pouvez activer la règle 123:11 afin de déclencher une alerte pour les fragments dont la TTL est inférieure à cette valeur.

Type : entier

Plage valide: de `1` à `255`

Valeur par défaut : `1`

politique

Spécifie le système d'exploitation des hôtes cibles. Le système d'exploitation détermine la politique de réassemblage des fragments IP et les caractéristiques du système d'exploitation. Vous ne pouvez définir qu'un seul paramètre `policy` pour chaque inspecteur de flux IP.

Remarque Si vous définissez le paramètre `policy` sur `first`, Snort peut offrir une protection partielle, mais risque de manquer des attaques. Vous devez modifier le paramètre `policy` de l'inspecteur de flux IP pour spécifier le système d'exploitation approprié.

Type : énumération

Valeurs valides : définissez un type de système d'exploitation pour le paramètre `policy`.

Tableau 30 : Valeurs valides pour le paramètre policy

Politique	Systèmes d'exploitation
`first`	Système d'exploitation inconnu
`linux`	Linux
`bsd`	AIX FreeBSD OpenBSD
`bsd_right`	HP JetDirect (imprimante)
`last`	Cisco IOS
`windows`	Windows 98 Windows NT Windows 2000 Windows XP
`solaris`	Système d'exploitation Cisco Solaris SunOS

Valeur par défaut : `linux`

session_timeout

Spécifie le nombre de secondes pendant lesquelles l'inspecteur `stream_ip` conserve un flux IP inactif dans la table d'état. Dès que Snort détecte un datagramme IP doté de la même clé de flux, il vérifie si la session du flux précédent a expiré. Si la session a expiré, Snort ferme le flux et en démarre un nouveau. Snort recherche les flux obsolètes associés à la configuration de flux de base.

Type : entier

Plage valide : de `0` à `2 147 483 647 (max31)`

Valeur par défaut : `60`

Règles de l'inspecteur de flux IP

Activez les règles de l'inspecteur `stream_ip` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 31 : Règles de l'inspecteur de flux IP

GID:SID	Message de règle
123:1	Options IP incohérentes sur les paquets fragmentés
123:2	Attaque Teardrop
123:3	Fragment court, tentative possible de déni de service (DoS)
123:4	Le paquet fragment se termine après le paquet défragmenté
123:5	Paquet fragment de taille nulle
123:6	Taille de fragment incorrecte, la taille du paquet est négative
123:7	Taille de fragment incorrecte, la taille du paquet est supérieure à 65 536
123:8	Chevauchement de fragmentation
123:11	Valeur TTL inférieure au minimum configuré, non utilisé pour le réassemblage
123:12	Chevauchement de fragment excessif
123:13	Fragment minuscule

Options des règles de prévention des intrusions de l'inspecteur de flux IP

L'inspecteur `stream_ip` ne comporte aucune option pour les règles de prévention des intrusions.

CHAPITRE 27

Inspecteur de flux TCP

Présentation de l'inspecteur de flux TCP

Type	Inspecteur (flux)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	Aucun
Activé	`vrai`

Le protocole TCP (Transmission Control Protocol) est un protocole de couche transport orienté connexion et avec état. TCP peut transmettre de manière fiable un flux ordonné d'octets entre un client et un serveur sur un réseau IP. TCP n'autorise qu'une seule connexion à la fois avec les mêmes valeurs de paramètres de connexion. Un système d'exploitation hôte gère les états d'une connexion TCP.

L'inspecteur `stream_tcp` assure le suivi, la normalisation et le réassemblage des flux TCP. Chaque inspecteur de flux TCP peut gérer le trafic TCP d'un ou plusieurs hôtes de votre réseau. En outre, si vous disposez de suffisamment d'informations sur les hôtes qui envoient le trafic TCP sur votre réseau, vous pouvez configurer un inspecteur `stream_tcp` pour ceux-ci.

Dans une politique d'analyse de réseau (NAP), Snort applique chaque inspecteur `stream_tcp` configuré aux services TCP définis dans la configuration de l'inspecteur `binder`.

Vous pouvez configurer plusieurs inspecteurs de flux TCP pour gérer différents systèmes d'exploitation et le trafic TCP.

La configuration de l'inspecteur `stream_tcp` comprend ce qui suit :

- Le système d'exploitation de l'hôte TCP

- Les options du système d'exploitation : mode de gestion des chevauchements pendant le réassemblage
- Les options de gestion du trafic : nombre maximal d'octets ou de segments dans une session ou direction
- Les options de réassemblage du flux TCP : taille maximale des PDU réassemblées

Remarque En mode IPS en ligne, l'inspecteur `stream_tcp` normalise le flux de la charge utile pour que les chevauchements soient toujours résolus en fonction de la première copie observée. Chaque inspecteur de flux TCP gère les SYN répétés, la validation RST et les vérifications d'horodatage.

Bonnes pratiques en matière de configuration de l'inspecteur de flux TCP

Tenez compte des bonnes pratiques suivantes lors de la configuration d'un inspecteur `stream_tcp` :

- Évitez de configurer les interfaces de détection sur un appareil pour que Snort ne puisse inspecter qu'un seul côté d'un flux. Vous pouvez activer le paramètre `reassemble_async` dans l'inspecteur `stream_tcp` pour traiter le trafic asymétrique. Cependant, l'inspecteur de flux TCP ne peut pas toujours traiter le trafic asymétrique. Par exemple, une réponse à une demande HTTP HEAD peut entraîner la désynchronisation de l'inspecteur HTTP. En mode IDS, l'absence d'accusés de réception TCP facilite les tentatives d'évasion.

  Pour le mode IPS, nous vous recommandons de ne déployer un appareil que si Snort peut inspecter les deux côtés d'un flux.

- Créez un inspecteur `stream_tcp` pour chaque système d'exploitation hôte TCP susceptible d'envoyer ou de recevoir du trafic TCP. Une même politique d'analyse de réseau peut contenir plusieurs versions de l'inspecteur `stream_tcp`. Les politiques TCP définies dans chaque inspecteur `stream_tcp` sont appliquées aux hôtes TCP spécifiés dans l'inspecteur `binder`.
- Pour activer le mode IPS, définissez le paramètre `normalizer.tcp.ips` de l'inspecteur `normalizer` sur `true`.
- Dans les paramètres avancés de votre politique d'analyse de réseau (NAP), confirmez que les réseaux que vous souhaitez identifier dans un inspecteur `stream_tcp` personnalisé basé sur des cibles correspondent à ou constituent un sous-ensemble des réseaux, des zones et des VLAN gérés par sa NAP parente.
- Le système crée une carte réseau distincte pour chaque domaine enfant. Dans un déploiement multidomaine, l'utilisation d'adresses IP littérales pour limiter cette configuration peut avoir des résultats inattendus. L'utilisation d'objets de substitution permet aux administrateurs de domaines descendants d'adapter les configurations globales à leurs environnements locaux.
- Pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés, activez les règles de l'inspecteur `stream_tcp` (GID 129).

Bonnes pratiques en matière de réassemblage de flux TCP

L'inspecteur `stream_tcp` collecte et réassemble tous les paquets qui font partie d'un flux de communication serveur-client, d'un flux de communication client-serveur ou des deux dans une session TCP. Le réassemblage du flux TCP permet à Snort d'inspecter le flux comme une seule unité réassemblée, appelée unité de données de protocole (PDU), plutôt que de se limiter à l'inspection des paquets individuels qui composent un flux donné. Si la PDU est volumineuse, le moteur de règles la divise en plusieurs parties.

Le réassemblage de flux permet à Snort d'identifier les attaques basées sur les flux, qu'il peut ne pas détecter lors de l'inspection de paquets individuels. Vous pouvez spécifier les flux de communication à réassembler en fonction des besoins de votre réseau. Par exemple, lorsque vous supervisez le trafic sur vos serveurs Web, il peut être pertinent de n'inspecter que le trafic client, car le risque de recevoir du trafic malveillant de votre propre serveur Web est généralement plus faible.

Pour chaque inspecteur `stream_tcp`, vous pouvez spécifier une liste de ports TCP dans la configuration du `binder`. L'inspecteur de flux TCP inclut automatiquement et de manière transparente les ports configurés pour identifier et réassembler le trafic. Si les mises à jour des profils adaptatifs sont activées, vous pouvez répertorier les services qui identifient le trafic à réassembler, soit comme alternative aux ports, soit en combinaison avec ceux-ci.

Spécifiez les ports TCP dans la configuration du `binder` pour les inspecteurs Snort suivants :

- `dnp3`
- `ftp_server`
- `gtp_inspect` (ports fournis par défaut)
- `http_inspect`
- `imap`
- `iec104` (ports fournis par défaut)
- `mms` (ports fournis par défaut)
- `modbus` (ports fournis par défaut)
- `pop`
- `sip`
- `smtp`
- `ssh`
- `ssl`
- `telnet`

Remarque Lorsque vous réassemblez plusieurs types de trafic (client, serveur, les deux), les demandes de ressources de Snort peuvent augmenter.

Paramètres de l'inspecteur de flux TCP

Configuration du réassemblage des flux TCP

L'inspecteur `binder` configure le réassemblage des flux TCP pour la politique d'analyse de réseau (NAP). Vous devez spécifier les adresses IP des hôtes auxquels vous souhaitez appliquer la politique de réassemblage des flux TCP. L'inspecteur de flux TCP est automatiquement associé aux ports configurés dans le `binder` pour la NAP. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Remarque Le système crée une carte réseau distincte pour chaque domaine enfant. Dans un déploiement multidomaine, l'utilisation d'adresses IP littérales pour limiter cette configuration peut avoir des résultats inattendus. L'utilisation d'objets de substitution permet aux administrateurs de domaines descendants d'adapter les configurations globales à leurs environnements locaux.

Le paramètre `default` de la politique par défaut spécifie toutes les adresses IP du segment de réseau supervisé qui ne sont couvertes par aucune autre politique basée sur la cible. Il est donc inutile et impossible de spécifier une adresse IP ou une longueur de bloc ou de préfixe CIDR pour la politique par défaut. De plus, vous ne pouvez pas laisser ce paramètre vide dans une autre politique ou utiliser la notation de l'adresse pour représenter `any` (par exemple, `0.0.0.0/0` ou `::/0`).

politique

Spécifie le système d'exploitation des hôtes cibles. Le système d'exploitation détermine la politique de réassemblage TCP et les caractéristiques du système d'exploitation. Vous ne pouvez définir qu'un seul paramètre `policy` pour chaque inspecteur de flux TCP.

Remarque Si vous définissez le paramètre `policy` sur `first`, Snort peut offrir une protection partielle, mais risque de manquer des attaques. Vous devez modifier le paramètre `policy` de l'inspecteur de flux TCP pour spécifier le système d'exploitation approprié.

Type : énumération

Valeurs valides : définissez un type de système d'exploitation pour le paramètre `policy`.

Tableau 32 : Politiques du système d'exploitation TCP

Politique	Systèmes d'exploitation
`first`	système d'exploitation inconnu
`last`	Cisco IOS
`bsd`	AIX FreeBSD OpenBSD

Politique	Systèmes d'exploitation
`hpux_10`	HP-UX 10.2 ou version antérieure
`hpux_11`	HP-UX 11.0 ou version ultérieure
`irix`	SGI Irix
`linux`	Noyau Linux 2.4 Noyau Linux 2.6
`macos`	Mac OS (Mac OS 10)
`old_linux`	Noyau Linux 2.2 et antérieur
`solaris`	Système d'exploitation Cisco Solaris SunOS
`vista`	Windows Vista
`windows`	Windows 98 Windows NT Windows 2000 Windows XP
`win_2003`	Windows 2003

Valeur par défaut : `bsd`

max_window

Spécifie la taille maximale autorisée par un hôte récepteur pour la fenêtre TCP. Vous pouvez spécifier un entier inférieur à `65 535`, ou `0` pour désactiver l'inspection de la taille de la fenêtre TCP.

Mise en garde La limite supérieure de `max_window` est la taille maximale de la fenêtre autorisée par la RFC 1323. Vous pouvez définir la limite supérieure pour empêcher un agresseur de se soustraire à la détection, mais une taille maximale de fenêtre TCP trop importante peut entraîner un déni de service auto-imposé.

Type : entier

Plage valide : de `0` à `1 073 725 440`

Valeur par défaut : `0`

overlap_limit

Spécifie le nombre maximal de segments autorisés à se chevaucher dans chaque session TCP. Spécifiez `0` pour autoriser le chevauchement d'un nombre illimité de segments. Si vous définissez un nombre compris entre `0` et `255`, le réassemblage des segments s'arrête pour la session.

Activez la règle 129:7 pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : 0

max_pdu

Spécifie la taille maximale des unités de données de protocole (PDU) réassemblées.

Type : entier

Plage valide : de `1 460` à `32 768`

Valeur par défaut : `16 384`

reassemble_async

Assure la mise en file d'attente des données pour le réassemblage préalable à la détection du trafic bidirectionnel. Lorsque le réseau supervisé est un réseau asynchrone, vous devez activer le paramètre `reassemble_async`. Dans un réseau asynchrone, le trafic est unidirectionnel et limité à un seul flux à la fois. L'activation du paramètre `reassemble_async` empêche Snort de réassembler les flux TCP, ce qui améliore les performances.

Remarque L'inspecteur de flux TCP ne peut pas toujours traiter le trafic asymétrique. Par exemple, une réponse à une demande HTTP HEAD peut entraîner la désynchronisation de l'inspecteur HTTP. En mode IDS, l'absence d'accusés de réception TCP facilite les tentatives d'évasion. Pour le mode IPS, nous vous recommandons de ne déployer un appareil que si le moteur de règles peut inspecter les deux côtés d'un flux.

Le paramètre `reassemble_async` est ignoré pour les interfaces Cisco Secure Firewall Threat Defense en mode routage et en mode transparent.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `true`

require_3whs

Spécifie le délai, en secondes après le démarrage, à l'issue duquel l'inspecteur de flux TCP cesse de suivre les sessions en cours. Spécifiez `-1` pour suivre toutes les sessions TCP en cours, quel que soit le moment où elles se produisent.

Snort ne synchronise pas la plupart des flux de protocoles. Snort procède toujours à la capture du SYN lorsqu'il a besoin d'une option d'établissement de liaison (horodatages, mise à l'échelle de fenêtre ou MSS). En règle générale, l'autorisation des captures en cours de transmission n'améliore pas l'efficacité de l'IPS.

Type : entier

Plage valide : de `-1` à `2 147 483 647 (max31)`

Valeur par défaut : `-1`

queue_limit.max_bytes

Spécifie le nombre maximal d'octets à mettre en file d'attente pour une session d'un côté d'une connexion TCP. Spécifiez `0` pour autoriser un nombre illimité d'octets.

Mise en garde Nous vous recommandons de contacter le centre d'assistance technique Cisco avant de modifier la valeur par défaut du paramètre `queue_limit.max_bytes`.

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `4 194 304`

queue_limit.max_segments

Spécifie le nombre maximal de segments de données à mettre en file d'attente pour une session d'un côté d'une connexion TCP. Spécifiez `0` pour autoriser un nombre illimité de segments de données.

Mise en garde Nous vous recommandons de contacter le centre d'assistance technique Cisco avant de modifier la valeur par défaut du paramètre `queue_limit.max_segments`.

Type : entier

Plage valide : de `0` à `4 294 967 295 (max32)`

Valeur par défaut : `3 072`

small_segments.count

Spécifie une valeur supérieure au nombre attendu de petits segments TCP consécutifs. Spécifiez `0` pour ignorer le nombre de petits segments TCP consécutifs.

Les paramètres `small_segments.count` et `small_segments.maximum_size` doivent être définis sur le même type de valeur. Spécifiez `0` pour les deux paramètres ou définissez chaque paramètre sur une valeur non nulle.

Remarque Snort considère 2 000 segments consécutifs, même si chacun ne fait que `1` octet, comme une quantité anormalement élevée de segments TCP consécutifs.

Snort ignore le paramètre `small_segments.count` pour les interfaces Threat Defence en mode routage et en mode transparent.

Vous pouvez activer la règle 129:12 pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Type : entier

Plage valide : de `0` à `2 048`

Valeur par défaut : `0`

small_segments.maximum_size

Spécifie le nombre d'octets qui définit un segment TCP comme étant plus grand qu'un petit segment TCP. La taille d'un petit segment TCP est comprise entre `1` et `2 048` octets. Spécifiez `0` pour ignorer la taille maximale d'un petit segment.

Snort ignore le paramètre `small_segments.maximum_size` pour les interfaces Threat Defence en mode routage et en mode transparent.

Les paramètres `small_segments.maximum_size` et `small_segments.count` doivent être définis sur le même type de valeur. Spécifiez `0` pour les deux paramètres ou définissez chaque paramètre sur une valeur non nulle.

Remarque Un segment TCP de `2 048` octets est plus grand qu'une trame Ethernet normale de `1 500` octets.

Vous pouvez activer la règle 129:12 pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Type : entier

Plage valide : de `0` à `2 048`

Valeur par défaut : `0`

session_timeout

Spécifie le nombre de secondes pendant lesquelles Snort conserve un flux TCP inactif dans sa table d'état. Si le flux n'est pas réassemblé dans le délai spécifié, Snort le supprime de la table d'état. Si la session est toujours active et que d'autres paquets apparaissent, Snort gère le flux comme un flux en cours.

Nous vous recommandons de définir le paramètre `session_timeout` sur une valeur supérieure ou égale au délai d'expiration de la session TCP de l'hôte.

Type : entier

Plage valide : de `0` à `2 147 483 647 (max31)`

Valeur par défaut : `180`

Règles de l'inspecteur de flux TCP

Activez les règles de l'inspecteur `stream_tcp` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 33 : Règles de l'inspecteur de flux TCP

GID:SID	Message de règle
129:1	SYN sur session établie
129:2	Données sur paquet SYN
129:3	Données envoyées sur un flux n'acceptant pas de données
129:4	Horodatage TCP en dehors de la fenêtre PAWS

GID:SID	Message de règle
129:5	Segment incorrect, taille ajustée <= 0 (obsolète)
129:6	Taille de la fenêtre (après mise à l'échelle) supérieure à la limite autorisée par la politique
129:7	Limite atteinte en termes de nombre de paquets TCP qui se chevauchent
129:8	Données envoyées sur un flux après l'envoi d'une réinitialisation TCP
129:9	Client TCP potentiellement piraté, adresse Ethernet différente
129:10	Serveur TCP potentiellement piraté, adresse Ethernet différente
129:11	Données TCP sans indicateur TCP défini
129:12	Segments TCP consécutifs de petite taille dépassant le seuil
129:13	Établissement de liaison en 4 étapes détectée
129:14	Horodatage TCP manquant
129:15	Réinitialisation en dehors de la fenêtre
129:16	Numéro de FIN supérieur au FIN précédent
129:17	Numéro de ACK supérieur au FIN précédent
129:18	Données envoyées sur un flux après réception d'une réinitialisation TCP
129:19	Fenêtre TCP fermée avant la réception de données
129:20	Session TCP sans établissement de liaison en 3 étapes

Options des règles de prévention des intrusions de l'inspecteur de flux TCP

stream_reassemble

Indiquez si le réassemblage des flux TCP doit être activé pour le trafic correspondant. L'option de règle `stream_reassemble` comprend quatre paramètres : `stream_reassemble.action`, `stream_reassemble.direction`, `stream_reassemble.noalert` et `stream_reassemble.fastpath`.

Syntaxe : `stream_reassemble: <enable|disable>, <server|client|both>, noalert, fastpath;`

Exemples : `stream_reassemble: disable,client,noalert;`

stream_reassemble.action

Arrêtez ou lancez le réassemblage du flux.

Type : énumération

Syntaxe : `stream_reassemble: <action>;`

Valeurs valides : `disable` ou `enable`

Exemples : `stream_reassemble: enable;`

stream_reassemble.direction

L'action s'applique aux sens donnés.

Type : énumération

Syntaxe : `stream_reassemble: <direction>`

Valeurs valides : `client`, `server`, `both`

Exemples : `stream_reassemble: both;`

stream_reassemble.noalert

Pas d'alerte en cas de correspondance de la règle. Le paramètre `stream_reassemble.noalert` est facultatif.

Syntaxe : `stream_reassemble: noalert;`

Exemples : `stream_reassemble: noalert;`

stream_reassemble.fastpath

Vous pouvez également approuver le reste de la session. Le paramètre `stream_reassemble.fastpath` est facultatif.

Syntaxe : `stream_reassemble: fastpath;`

Exemples : `stream_reassemble: fastpath;`

stream_size

Option de détection pour la vérification de la taille des flux. Permet à une règle de trouver une correspondance avec le trafic en fonction du nombre d'octets observés, comme déterminé par les numéros de séquence TCP. L'option de règle `stream_size` comprend deux paramètres : `stream_size.direction` et `stream_size.range`.

Syntaxe : `stream_size: <server|client|both|either>, <operator><number>;`

Exemples : `stream_size: client, <6;`

stream_size.direction

La comparaison s'applique au sens du flux.

Type : énumération

Syntaxe : `stream_size: <direction>;`

Valeurs valides :

- `either`
- `to_server`
- `to_client`
- `both`

Exemples : `stream_size: to_client;`

stream_size.range

Déterminez si la taille du flux se situe dans la plage spécifiée. Spécifiez un opérateur `range` et un ou plusieurs entiers positifs.

Type : intervalle

Syntaxe : `stream_size: <range_operator><positive integer>;` ou `stream_size: <positive integer><range_operator><positive integer>;`

Valeurs valides : un ensemble d'un ou de plusieurs entiers positifs, et un opérateur `range_operator`, comme spécifié dans le Tableau 34 : Formats de plages.

Exemples : `stream_size: >6;`

Tableau 34 : Formats de plages

Format de plage	Opérateur	Description
opérateur i		
	<	Supérieur à
	>	Supérieur à
	=	Égal à
	≠	Différent de
	≤	Inférieur ou égal à
	≥	Supérieur ou égal à
j opérateur k		
	<>	Supérieur à j et inférieur à k
	<=>	Supérieur ou égal à j et inférieur ou égal à k

CHAPITRE 28

Inspecteur de flux UDP

Présentation de l'inspecteur de flux UDP

Type	Inspecteur (flux)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	Aucun
Activé	`vrai`

Le protocole UDP (User Datagram Protocol) est un protocole de couche transport sans connexion et à faible latence. UDP permet une communication sans état entre deux terminaux du réseau avant que la partie réceptrice ne donne son accord. Pour évaluer l'intégrité de l'en-tête et des données du message, UDP utilise des sommes de contrôle.

L'inspecteur `stream_udp` vérifie les champs d'adresse IP source et de destination dans l'en-tête du datagramme IP ainsi que les champs de port dans l'en-tête UDP pour déterminer le sens du flux et identifier une session. Une session se termine lorsqu'un délai configurable est dépassé ou lorsqu'un terminal reçoit un message ICMP indiquant que l'autre terminal est inaccessible.

L'inspecteur de flux UDP ne génère pas d'événements. Vous pouvez activer les règles du décodeur de paquets (GID 116) pour détecter les anomalies au sein des en-têtes UDP.

Bonnes pratiques en matière de configuration de l'inspecteur de flux UDP

Tenez compte des bonnes pratiques suivantes lors de la configuration de l'inspecteur `stream_udp` :

- Créez un inspecteur `stream_udp` pour chaque délai d'expiration de session que vous souhaitez appliquer à un hôte ou à un terminal. L'inspecteur de flux UDP associe le paramètre `session_timeout` aux hôtes UDP définis dans l'inspecteur `binder`.

  Une même politique d'analyse de réseau peut contenir plusieurs versions de l'inspecteur `stream_udp`.

- Activez les règles du décodeur de paquets (GID 116) pour détecter les anomalies au sein des en-têtes UDP.

Paramètres de l'inspecteur de flux UDP

session_timeout

Spécifie le nombre de secondes pendant lesquelles l'inspecteur UDP conserve un flux UDP inactif dans la table d'état. Dès que Snort détecte un datagramme UDP doté de la même clé de flux, il vérifie si la session du flux précédent a expiré. Si la session a expiré, Snort ferme le flux et en démarre un nouveau. Snort recherche les flux obsolètes associés à la configuration de flux de base.

Type : entier

Plage valide : de `0` à `2 147 483 647 (max31)`

Valeur par défaut : `30`

Règles de l'inspecteur de flux UDP

Aucune règle n'est associée à l'inspecteur `stream_udp`.

Options des règles de prévention des intrusions de l'inspecteur de flux UDP

L'inspecteur `stream_udp` ne comporte aucune option pour les règles de prévention des intrusions.

CHAPITRE 29

Inspecteur Telnet

Présentation de l'inspecteur Telnet

Type	Inspecteur (service)
Usage	Inspecter
Type d'instance	Multiton
Autres inspecteurs requis	`stream_tcp`
Activé	`faux`

Telnet est un protocole de couche application qui crée un canal de communication en octets de 8 bits sur TCP. Telnet utilise un terminal virtuel réseau pour établir la communication entre un client et un hôte distant. Un serveur Telnet utilise le port TCP 23.

L'inspecteur `telnet` normalise le tampon de données Telnet en détectant les séquences de commandes Telnet et la négociation des options. L'inspecteur `telnet` élimine les séquences de commandes Telnet (RFC 854) du paquet. L'inspecteur `telnet` peut détecter les connexions Telnet chiffrées en examinant l'option de chiffrement Telnet (RFC 2946).

Paramètres de l'inspecteur Telnet

Configuration du service Telnet

L'inspecteur `binder` configure le `service` Telnet. Pour obtenir plus d'informations, reportez-vous à la Présentation de l'inspecteur de binder, à la page 15.

Exemple :

```
[
    {
        "when": {
            "service": "telnet",
            "role": any
        },
        "use": {
            "type": "telnet"
        }
    }
]
```

ayt_attack_thresh

Spécifie le nombre maximal de commandes Telnet AYT (Are You There) consécutives. L'inspecteur `telnet` détecte et génère des alertes sur le nombre de commandes AYT consécutives qui dépassent la valeur `ayt_attack_thresh`. Le paramètre `ayt_attack_thresh` vise à contrer des vulnérabilités spécifiques aux implémentations BSD de Telnet. Spécifiez `-1` pour désactiver le paramètre `ayt_attack_thresh`. En ce qui concerne ce paramètre, vous pouvez activer la règle 126:1 pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Type : entier

Plage valide : de `-1` à `2 147 483 647 (max31)`

Valeur par défaut : `-1`

encrypted_traffic

Indique si le trafic Telnet chiffré doit être détecté. En ce qui concerne ce paramètre, vous pouvez activer la règle 126:2 pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

normalize

Indique si le trafic Telnet doit être normalisé. L'inspecteur `Telnet` normalise le trafic Telnet en éliminant les séquences d'échappement Telnet. Si une règle de prévention des intrusions activée spécifie un paramètre de contenu `raw`, la règle ignore le tampon Telnet normalisé créé par l'inspecteur `telnet`.

Type : booléen

Valeurs valides : `true`, `false`

Valeur par défaut : `false`

Règles de l'inspecteur Telnet

Activez l'inspecteur `telnet` pour générer des événements et, dans un déploiement en ligne, supprimer les paquets incriminés.

Tableau 35 : Règles de l'inspecteur Telnet

GID:SID	Message de règle
126:1	Commandes Telnet AYT consécutives au-delà du seuil
126:2	Trafic Telnet chiffré
126:3	Commande de début de sous-négociation Telnet sans fin de sous-négociation correspondante

Options des règles de prévention des intrusions de l'inspecteur Telnet

L'inspecteur `telnet` ne comporte aucune option pour les règles de prévention des intrusions.

À propos de la traduction

Cisco peut fournir des traductions du présent contenu dans la langue locale pour certains endroits. Veuillez noter que des traductions sont fournies à titre informatif seulement et, en cas d’incohérence, la version anglaise du présent contenu prévaudra.