Référence pour les contrôles de conformité Nessus

10 janvier 2014

(Révision 39)

Table des matières

Introduction

Ce document décrit la syntaxe utilisée pour créer des fichiers personnalisés **.audit** qui peuvent être employés pour vérifier la configuration des systèmes Unix, Windows, base de données, SCADA, IBM iSeries et Cisco par rapport à une stratégie de conformité et pour rechercher un contenu spécifique dans le contenu de divers systèmes.

Ce guide est conçu pour faciliter la création manuelle et la compréhension de la syntaxe des fichiers d'audit de conformité. Veuillez vous reporter au PDF des contrôles de conformité Nessus disponible sur le portail d'assistance de Tenable (Tenable Support Portal) pour en savoir plus sur le fonctionnement des contrôles de conformité Tenable.

Nessus prend en charge l'audit du système SCADA ; toutefois, cette fonctionnalité n'est pas abordée dans ce document. Voir la page d'information Tenable SCADA ici pour plus d'informations.

Prérequis

Ce document suppose un certain niveau de connaissances sur le scanner de vulnérabilité Nessus ainsi qu'une compréhension détaillée des systèmes cible vérifiés. Pour de plus amples renseignements sur la façon dont Nessus peut être configuré pour effectuer des audits de correctif Unix et Windows locaux, voir le document « Nessus Credentials Checks for Unix and Windows » (Contrôles des identifiants Nessus pour Unix et Windows) disponible sur http://www.tenable.com/products/nessus/documentation.

Normes et conventions

Dans l'ensemble de la documentation, les noms de fichiers, les démons (daemons) et les exécutables sont indiqués par la police **courier bold**.

Les options de ligne de commande et les mots clés sont aussi indiqués par la police **courier bold**. Les exemples de ligne de commande peuvent inclure ou non l'invite de ligne de commande et le texte provenant des résultats de la commande. Les exemples de ligne de commande sont affichés en **courier bold** dans la commande en cours d'exécution afin de montrer la saisie de l'utilisateur, tandis que l'exemple de sortie généré par le système utilisera la police courier (mais pas en gras). Voici ci-dessous un exemple d'exécution de la commande Unix **pwd** :

```
# pwd
/home/test/
#
```

Les remarques et considérations importantes sont mises en évidence avec ce symbole dans une boîte de texte grise.

Les conseils, exemples et meilleures pratiques sont mis en évidence avec ce symbole en texte blanc sur fond bleu.

Référence des fichiers de conformité d'audit pour la configuration de Windows

La base des fichiers de conformité Windows **.audit** est un fichier de texte spécialement formaté. Les entrées du fichier peuvent invoquer divers contrôles d'« éléments personnalisés » tels que les contrôles de définition de registre, ainsi que des contrôles plus génériques tels que les contrôles de définition des stratégies locales de sécurité. Des exemples sont utilisés dans l'ensemble du guide par souci de clarté.

Emploi des guillemets :

Les guillemets simples et doubles sont interchangeables autour des champs d'audit, sauf dans les deux cas suivants :

1. Dans les contrôles de conformité Windows où des champs spéciaux, tels que CRLF, doivent être interprétés littéralement, utilisez des guillemets simples. Tout champ intégré qui doit être interprété comme une chaîne doit être inclus dans une séquence d'échappement.

Par exemple :
```
expect: 'First line\r\nSecond line\r\nJohn\'s Line'
```

2. Des guillemets doubles sont requis lorsque vous utilisez les fichiers Windows « include_paths » et « exclude_paths ».

Si des chaînes sont utilisées dans tout type de champ (description, value_data, regex, etc.) contenant des guillemets simples ou doubles, il y a deux façons de les traiter :

a. Utilisez le type de guillemets opposés pour les guillemets extérieurs.

Par exemple :
```
expect: "This is John's Line"
expect: 'We are looking for a double-quote-".*'
```

b. Échappez tout guillemet intégré avec une barre oblique inverse (guillemets doubles uniquement).

Par exemple :
```
expect: "\"Text to be searched\""
```

Type de contrôle

Tous les contrôles de conformité Windows doivent être mis entre crochets avec l'encapsulation **`check_type`** avec désignation « Windows », et la version « 2 » doit être précisée :

```
<check_type:"Windows" version:"2">
```

Un exemple de contrôle de conformité Windows est indiqué dans l'Annexe B, commençant par le paramètre **`check_type`** pour « Windows » et la version « 2 » et se terminant par la balise « **`</check_type>`** ».

Ceci est requis pour distinguer les fichiers Windows **`.audit`** de ceux destinés à Unix (ou à d'autres plateformes).

Données de valeur

La syntaxe du fichier **`.audit`** contient des mots clés auxquels divers types de valeur peuvent être attribués pour personnaliser les contrôles. Cette section décrit ces mots clés et le format des données qui peuvent être saisies.

Types de données

Les types de données ci-dessous peuvent être saisis pour les contrôles :

Type de données	Description
DWORD	0 à 2,147,483,647
PLAGE [X..Y]	Où X est un DWORD ou MIN et Y est un DWORD ou MAX

Exemples :

```
value_data: 45
value_data: [11..9841]
value_data: [45..MAX]
```

En outre, les nombres peuvent être spécifiés avec plus (+) ou moins (-) pour indiquer leur « signe » et ils peuvent être spécifiés comme étant des valeurs hexadécimales. Les hexadécimaux et les signes peuvent être combinés. Les hexadécimaux et les signes peuvent être combinés. Des exemples valides (sans la balise correspondante entre parenthèses) sont indiqués ci-dessous dans le cadre d'un audit REGISTRY_SETTING pour un POLICY_DWORD :

```
value_data: -1 (signed)
value_data: +10 (signed)
value_data: 10 (unsigned)
value_data: 2401649476 (unsigned)
value data: [MIN..+10] (signed range)
value_data: [20..MAX] (unsigned range)
value_data: 0x800010AB (unsigned hex)
value_data: -0x10 (signed hex)
```

Expressions complexes

Les expressions complexes peuvent être utilisées pour le champ **`value_data`** en utilisant :

- || : OU conditionnel
- && : ET conditionnel
- | : OU binaire (opération sur bits)
- & : ET binaire (opération sur bits)
- (and) : pour délimiter les expressions complexes

Exemples :

```
value data: 45 || 10
value_data: (45 || 10) && ([9..12] || 37)
```

Champ « check_type » (type de contrôle)

Ce type de contrôle est différent du champ « **`check_type`** » spécifié ci-dessus qui est utilisé au début de chaque fichier d'audit pour indiquer le type générique d'audit (Windows, WindowsFiles, Unix, Database, Cisco). Il est facultatif et peut être effectué en fonction de valeurs Windows **`value_data`** pour déterminer le type de contrôle à exécuter. Les paramètres suivants sont disponibles :

- CHECK_EQUAL : compare la valeur à distance à la valeur de la stratégie (par défaut si **`check_type`** est absent)
- CHECK_EQUAL_ANY : vérifie que chaque élément de **`value_data`** est présent au moins une fois dans la liste du système
- CHECK_NOT_EQUAL : vérifie que la valeur à distance est différente de la valeur de la stratégie
- CHECK_GREATER_THAN : vérifie que la valeur à distance est supérieure à la valeur de la stratégie

- CHECK_GREATER_THAN_OR_EQUAL : vérifie que la valeur à distance est supérieure ou égale à la valeur de la stratégie
- CHECK_LESS_THAN : vérifie que la valeur à distance est inférieure à la valeur de la stratégie
- CHECK_LESS_THAN_OR_EQUAL : vérifie que la valeur à distance est inférieure ou égale à la valeur de la stratégie
- CHECK_REGEX : vérifie que la valeur à distance correspond au regex dans la valeur de la stratégie (fonctionne seulement avec POLICY_TEXT et POLICY_MULTI_TEXT)
- CHECK_SUBSET : vérifie que l'ACL à distance est un sous-ensemble de l'ACL de la stratégie (fonctionne seulement avec les ACL)
- CHECK_SUPERSET : vérifie que l'ACL à distance est un sur-ensemble de l'ACL de la stratégie (fonctionne seulement avec les ACL de refus de droits)

L'exemple d'audit ci-dessous vérifie que le nom de compte « Guest » (Invité) n'existe pour aucun compte d'invité.

```
<custom_item>
 type: CHECK_ACCOUNT
 description: "Accounts: Rename guest account"
 value_type: POLICY_TEXT
 value_data: "Guest"
 account_type: GUEST_ACCOUNT
 check_type: CHECK_NOT_EQUAL
</custom_item>
```

Si une valeur autre que « Guest » (Invité) est présente, le test réussit. Si « Guest » (Invité) est trouvé, l'audit échoue.

Champ « group_policy » (stratégie de groupe)

Le champ « **group_policy** » peut être utilisé pour fournir une courte chaîne de texte servant à décrire l'audit. Le champ **group_policy** doit être inclus dans un fichier d'audit et inséré après le champ **check_type**.

```
<check_type: "Windows" version:"2">
<group_policy: "Audit file for Windows 2008">

…

</group_policy
</check_type>
```

Champ « info » (info)

Le champ facultatif « **info** » peut être utilisé pour étiqueter chaque champ d'audit avec une ou plusieurs références externes. Par exemple, ce champ sera utilisé pour placer des références des balises NIST CCE ainsi que des exigences d'audit CIS spécifiques. Ces références externes sont imprimées dans l'audit final effectué par Nessus et seront indiquées dans le rapport Nessus ou par l'interface d'utilisateur de SecurityCenter.

La stratégie d'audit de mot de passe ci-dessous a été augmentée pour répertorier les références à une stratégie d'entreprise fictive :

```
<custom_item>
 type: PASSWORD_POLICY
 description: "Password History: 24 passwords remembered"
```

```
 value_type: POLICY_DWORD
 value_data: [22..MAX] || 20
 password_policy: ENFORCE_PASSWORD_HISTORY
 info: "Corporate Policy 102-A"
</custom_item>
```

Si plusieurs références de stratégie sont requises pour un seul audit, la chaîne spécifiée par le mot clé « **info** » peut utiliser le séparateur « \n » pour spécifier plusieurs chaînes. Par exemple, considérez l'audit suivant :

```
<custom_item>
 type: CHECK_ACCOUNT
 description: "Accounts: Rename Administrator account"
 value_type: POLICY_TEXT
 value_data: "Administrator"
 account_type: ADMINISTRATOR_ACCOUNT
 check_type: CHECK_NOT_EQUAL
 info: 'Ron Gula Mambo Number 5\nCCE-60\nTenable Best Practices Policy 1005-a'
</custom_item>
```

Lorsqu'elle est exécutée avec l'outil de ligne de commande **nasl**, cette fonction de vérification produit la sortie suivante :

```
# /opt/nessus/bin/nasl -t 192.168.20.16 ./compliance_check.nbin

Windows Compliance Checks, version 2.0.0

Which file contains your security policy : ./test_v2.audit
SMB login : Administrator
SMB password :
SMB domain (optional) :
"Accounts: Rename Administrator account": [FAILED]

Ron Gula Mambo Number 5
CCE-60
Tenable Best Practices Policy 1005-a

Remote value: "Administrator"
Policy value: "administrator"
```

Champ « debug » (débogage)

Le champ « **debug** » facultatif peut être utilisé pour résoudre les problèmes des contrôles de conformité du contenu Windows. Le mot clé debug permet d'obtenir des informations sur le scan de contenu en cours d'exécution, par exemple le(s) fichier(s) en cours de traitement, de scan et la détection ou non de résultats. En raison du grand nombre de résultats, ce mot clé ne doit être utilisé qu'à des fins de résolution des problèmes. Par exemple :

```
<item>
 debug
 type: FILE_CONTENT_CHECK
 description: "TNS File that Contains the word Nessus"
 file_extension: "tns"
 expect: "Nessus"
</item>
```

Format ACL

Cette section décrit la syntaxe utilisée pour déterminer si un fichier ou un dossier possède le paramètre ACL souhaité.

Vérifications de contrôle d'accès aux fichiers

Utilisation
<file_acl: ["name"]> <user: ["user_name"]> acl_inheritance: ["value"] acl_apply: ["value"] *(optional)* acl_allow: ["rights value"] *(optional)* acl_deny: ["rights value"] </user> </acl>

Une ACL (Access Control List, liste de contrôle d'accès) de fichier est identifiée par le mot clé **`file_acl`**. Le nom de l'ACL doit être unique pour être utilisé avec un élément de permissions de fichier. Une ACL de fichier peut contenir une ou plusieurs entrées d'utilisateur.

Types associés	Types autorisés
`acl_inheritance`	• not inherited (non hérité) • inherited (hérité) • not used (non utilisé)
`acl_apply`	• this folder only (ce dossier seulement) • this object only (cet objet seulement) • this folder and files (ce dossier et ses fichiers) • this folder and subfolders (ce dossier et ses sous-dossiers) • this folder, subfolders and files (ce dossier, ses sous-dossiers et ses fichiers) • files only (fichiers seulement) • subfolders only (sous-dossiers seulement) • subfolders and files only (sous-dossiers et fichiers seulement)
`acl_allow` **`acl_deny`**	Ces paramètres sont facultatifs. Les droits génériques sont : • full control (contrôle complet) • modify (modifier) • read & execute (lire et exécuter) • read (lire) • write (écrire) • list folder contents (répertorier le contenu des dossiers) Les droits avancés sont : • full control (contrôle complet) • traverse folder / execute file (défiler dossier / exécuter fichier) • list folder / read data (répertorier dossier / lire données) • read attributes (lire attributs) • read extended attributes (lire attributs étendus)

	• create files / write data (créer fichier / écrire données) • create folders / append data (créer dossiers / ajouter données) • write attributes (écrire attributs) • write extended attributes (écrire attributs étendus) • delete subfolder and files (supprimer sous-dossier et fichiers) • delete (supprimer) • read permissions (lire permissions) • change permissions (changer permissions) • take ownership (assumer propriété)

Voici un exemple de texte de contrôle d'accès de fichier **`.audit`** :

```
 <file_acl: "ASU1">

  <user: "Administrators">
   acl_inheritance: "not inherited"
   acl_apply: "This folder, subfolders and files"
   acl_allow: "Full Control"
  </user>

  <user: "System">
   acl_inheritance: "not inherited"
   acl_apply: "This folder, subfolders and files"
   acl_allow: "Full Control"
  </user>

  <user: "Users">
   acl_inheritance: "not inherited"
   acl_apply: "this folder only"
   acl_allow: "list folder / read data" | "read attributes" | "read extended
        attributes" | "create files / write data" | "create folders / append data" |
        "write attributes" | "write extended attributes" | "read permissions"
  </user>

 </acl>
```

Vérifications de contrôle d'accès au registre

Utilisation

```
<registry_acl: ["name"]>

 <user: ["user_name"]>
  acl_inheritance: ["value"]
  acl_apply: ["value"]
  (optional) acl_allow: ["rights value"]
  (optional) acl_deny: ["rights value"]
 </user>

</acl>
```

Une ACL de registre est identifiée par le mot clé **`registry_acl`**. Le nom de l'ACL doit être unique pour être utilisé avec un élément de permissions de registre. Une ACL de registre peut contenir une ou plusieurs entrées d'utilisateur.

Types associés	Types autorisés
acl_inheritance	• not inherited (non hérité) • inherited (hérité) • not used (non utilisé)
acl_apply	• this key only (cette clé seulement) • this key and subkeys (cette clé et ses sous-clés) • subkeys only (sous-clés seulement)
acl_allow **acl_deny**	Ces paramètres sont facultatifs et sont utilisés pour définir les droits détenus par un utilisateur sur l'objet. Les droits génériques sont : • full control (contrôle complet) • read (lire) Les droits avancés sont : • full control (contrôle complet) • query value (valeur d'interrogation) • set value (valeur définie) • create subkey (créer sous-clé) • enumerate subkeys (énumérer sous-clés) • notify (notifier) • create link (créer lien) • delete (supprimer) • write dac (écrire dac) • write owner (écrire propriétaire) • read control (contrôle de lecture)

Voici un exemple de texte de liste de contrôle d'accès de registre **.audit** :

```
<registry_acl: "SOFTWARE ACL">

 <user: "Administrators">
  acl_inheritance: "not inherited"
  acl_apply: "This key and subkeys"
  acl_allow: "Full Control"
 </user>

 <user: "CREATOR OWNER">
  acl_inheritance: "not inherited"
  acl_apply: "Subkeys only"
  acl_allow: "Full Control"
 </user>

 <user: "SYSTEM">
  acl_inheritance: "not inherited"
  acl_apply: "This key and subkeys"
  acl_allow: "Full Control"
 </user>
```

```
  <user: "Users">
   acl_inheritance: "not inherited"
   acl_apply: "This key and subkeys"
   acl_allow: "Read"
  </user>

 </acl>
```

Vérifications de contrôle d'accès au service

Utilisation

`<service_acl: ["name"]>`
 	
` <user: ["user_name"]>`
`  acl_inheritance: ["value"]`
`  acl_apply: ["value"]`
`  (optional) acl_allow: ["rights value"]`
`  (optional) acl_deny: ["rights value"]`
` </user>`
 	
`</acl>`

Une ACL de service est identifiée par le mot clé **`service_acl`**. Le nom de l'ACL doit être unique pour être utilisé avec un élément de permissions de service. Une ACL de service peut contenir une ou plusieurs entrées d'utilisateur.

Types associés	Types autorisés
`acl_inheritance`	• not inherited (non hérité) • inherited (hérité) • not used (non utilisé)
`acl_apply`	• this object only (cet objet seulement)
`acl_allow` **`acl_deny`**	Ces paramètres sont facultatifs et sont utilisés pour définir les droits détenus par un utilisateur sur l'objet. Les droits génériques sont : • full control (contrôle complet) • read (lire) • start, stop and pause (démarrer, arrêter et faire une pause) • write (écrire) • delete (supprimer) Les droits avancés sont : • full control (contrôle complet) • delete (supprimer) • query template (questionner modèle) • change template (changer modèle) • query status (questionner état)

	• enumerate dependents (énumérer dépendants) • start (démarrer) • stop (arrêter) • pause and continue (faire une pause et continuer) • interrogate (interroger) • user-defined control (contrôle défini par l'utilisateur) • read permissions (lire permissions) • change permissions (changer permissions) • take ownership (assumer propriété)

Voici un exemple de vérification de contrôle d'accès au service :

```
 <service_acl: "ALERT ACL">

  <user: "Administrators">
   acl_inheritance: "not inherited"
   acl_apply: "This object only"
   acl_allow: "query template" | "change template" | "query status" | "enumerate
        dependents" | "start" | "stop" | "pause and continue" | "interrogate" | "user-
        defined control" | "delete" | "read permissions" | "change permissions" | "take
        ownership"
  </user>

 </acl>
```

Vérifications de contrôle de permission de lancement

Utilisation

```
<launch_acl: ["name"]>

 <user: ["user_name"]>
  acl_inheritance: ["value"]
  acl_apply: ["value"]
  (optional) acl_allow: ["rights value"]
  (optional) acl_deny: ["rights value"]
 </user>

</acl>
```

Une launch ACL (ACL de lancement) est identifiée par le mot clé **`launch_acl`**. Le nom de l'ACL doit être unique pour être utilisé avec un élément de permissions de lancement DCOM. Une ACL de lancement peut contenir une ou plusieurs entrées d'utilisateur.

Types associés	Types autorisés
`acl_inheritance`	• not inherited (non hérité) • inherited (hérité)
`acl_apply`	• this object only (cet objet seulement)

acl_allow **acl_deny**	Ces paramètres sont facultatifs et sont utilisés pour définir les droits détenus par un utilisateur sur l'objet. Les droits génériques sont : • local launch (lancement local) • remote launch (lancement à distance) • local activation (activation locale) • remote activation (activation à distance)

Cette ACL fonctionne seulement avec Windows XP/2003/Vista (et partiellement avec Windows 2000).

Voici un exemple de vérification de contrôle d'accès de lancement :

```
 <launch_acl: "2">

  <user: "Administrators">
   acl_inheritance: "not inherited"
   acl_apply: "This object only"
   acl_allow: "Remote Activation"
  </user>

  <user: "INTERACTIVE">
   acl_inheritance: "not inherited"
   acl_apply: "This object only"
   acl_allow: "Local Activation" | "Local Launch"
  </user>

  <user: "SYSTEM">
   acl_inheritance: "not inherited"
   acl_apply: "This object only"
   acl_allow: "Local Activation" | "Local Launch"
  </user>

 </acl>
```

Vérifications de contrôle de permission Launch2

Utilisation

```
<launch2_acl: ["name"]>

 <user: ["user_name"]>
  acl_inheritance: ["value"]
  acl_apply: ["value"]
  (optional) acl_allow: ["rights value"]
  (optional) acl_deny: ["rights value"]
 </user>

</acl>
```

Une ACL launch2 est identifiée par le mot clé **`launch2_acl`**. Le nom de l'ACL doit être unique pour être utilisé avec un élément de permissions de lancement DCOM. Une ACL launch2 peut contenir une ou plusieurs entrées d'utilisateur.

Types associés	Types autorisés
`acl_inheritance`	• not inherited (non hérité) • inherited (hérité)
`acl_apply`	• this object only (cet objet seulement)
`acl_allow` **`acl_deny`**	Ces paramètres sont facultatifs et sont utilisés pour définir les droits détenus par un utilisateur sur l'objet. Les droits génériques sont : • launch (lancement)

Utilisez seulement l'ACL launch2 avec les systèmes Windows 2000 et NT.

Voici un exemple de vérification de contrôle d'accès de lancement :

```
<launch2_acl: "2">

 <user: "Administrators">
  acl_inheritance: "not inherited"
  acl_apply: "This object only"
  acl_allow: "Launch"
 </user>

 <user: "INTERACTIVE">
  acl_inheritance: "not inherited"
  acl_apply: "This object only"
  acl_allow: "Launch"
 </user>

 <user: "SYSTEM">
  acl_inheritance: "not inherited"
  acl_apply: "This object only"
  acl_allow: "Launch"
 </user>

</acl>
```

Vérifications de contrôle de permission d'accès

Utilisation

`<access_acl: ["name"]>`
 	
`<user: ["user_name"]>`

```
  acl_inheritance: ["value"]
  acl_apply: ["value"]
  (optional) acl_allow: ["rights value"]
  (optional) acl_deny: ["rights value"]
 </user>

</acl>
```

Une ACL d'accès est identifiée par le mot clé **`access_acl`**. Le nom de l'ACL doit être unique pour être utilisé avec un élément de permissions d'accès DCOM. Une ACL d'accès peut contenir une ou plusieurs entrées d'utilisateur.

Types associés	Types autorisés
`acl_inheritance`	• not inherited (non hérité) • inherited (hérité)
`acl_apply`	• this object only (cet objet seulement)
`acl_allow` **`acl_deny`**	Ces paramètres sont facultatifs et sont utilisés pour définir les droits détenus par un utilisateur sur l'objet. Les droits génériques sont : • local access (accès local) • remote access (accès à distance)

Voici un exemple de vérification de contrôle d'accès :

```
 <access_acl: "3">

  <user: "SELF">
   acl_inheritance: "not inherited"
   acl_apply: "This object only"
   acl_allow: "Local Access"
  </user>

  <user: "SYSTEM">
   acl_inheritance: "not inherited"
   acl_apply: "This object only"
   acl_allow: "Local Access"
  </user>

  <user: "Users">
   acl_inheritance: "not inherited"
   acl_apply: "This object only"
   acl_allow: "Local Access"
  </user>

 </acl>
```

Éléments personnalisés

Un élément personnalisé est un contrôle complet défini en fonction des mots clés établis ci-dessus. Une liste de types d'éléments personnalisés disponibles est indiquée ci-dessous. Chaque contrôle commence par une balise

« **<custom_item>** » et se termine par « **</custom_item>** ». Les balises contiennent des listes d'un ou plusieurs mots clés qui sont interprétés par l'analyseur syntaxique de contrôle de conformité pour effectuer les contrôles.

Les contrôles d'audit personnalisé peuvent utiliser indifféremment « **</custom_item>** » et « **</item>** » pour la balise de fermeture.

PASSWORD_POLICY

Utilisation

```
<custom_item>
 type: PASSWORD_POLICY
 description: ["description"]
 value type: [VALUE TYPE]
 value_data: [value]
 (optional) check_type: [value]
 password_policy: [PASSWORD_POLICY_TYPE]
</custom_item>
```

Cet élément de stratégie contrôle les valeurs définies dans « Paramètres Windows -> Paramètres de sécurité -> Stratégies de comptes -> Stratégie de mot de passe ».

Le contrôle est effectué en invoquant la fonction **NetUserModalsGet** avec le niveau 1.

Ces éléments utilisent le champ **password_policy** pour décrire l'élément de la stratégie de mot de passe qui doit être vérifié. Les types autorisés sont :

- ENFORCE_PASSWORD_HISTORY ("Enforce password history"- Appliquer l'historique des mots de passe)
  `value_type: POLICY_DWORD`
  `value_data: DWORD ou RANGE [nombre de mots de passe mémorisés]`
- MAXIMUM_PASSWORD_AGE ("Maximum password age"- Antériorité maximale du mot de passe)
  `value_type: TIME_DAY`
  `value_data: DWORD ou RANGE [durée en jours]`
- MINIMUM_PASSWORD_AGE ("Maximum password age"- Antériorité minimale du mot de passe)
  `value_type: TIME_DAY`
  `value_data: DWORD ou RANGE [durée en jours]`
- MINIMUM_PASSWORD_LENGTH ("Minimum password length" - Longueur minimale du mot de passe)
  `value_type: POLICY_DWORD`
  `value_data: DWORD ou RANGE [nombre minimal de caractères du mot de passe]`
- COMPLEXITY_REQUIREMENTS ("Password must meet complexity requirements" - Le mot de passe doit respecter des exigences de complexité)
  `value_type: POLICY_SET`
  `value_data: "Enabled" ou "Disabled"`
- REVERSIBLE_ENCRYPTION ("Store passwords using reversible encryption for all users in the domain" - Stocker les mots de passe en utilisant le cryptage réversible pour tous les utilisateurs du domaine)
  `value_type: POLICY_SET`
  `value_data: "Enabled" ou "Disabled"`

- FORCE_LOGOFF ("Network security: Force logoff when logon hours expire" - Sécurité réseau : forcer la fermeture de session quand les horaires de connexion expirent)
  `value_type: POLICY_SET`
  `value_data: "Enabled" ou "Disabled"`

Il n'existe actuellement aucune façon de contrôler la stratégie « Store password using reversible encryption for all users in the domain » (Stocker les mots de passe en utilisant le cryptage réversible pour tous les utilisateurs du domaine).

La stratégie FORCE_LOGOFF est située dans « Paramètres de sécurité -> Stratégies locales -> Options de sécurité ».

Voici un exemple d'audit de stratégie du mot de passe :

```
<custom_item>
 type: PASSWORD_POLICY
 description: "Minimum password length"
 value_type: POLICY_DWORD
 value_data: 7
 password_policy: MINIMUM_PASSWORD_LENGTH
</custom_item>
```

LOCKOUT_POLICY

Utilisation

```
<custom_item>
 type: LOCKOUT_POLICY
 description: ["description"]
 value_type: [VALUE_TYPE]
 value_data: [value]
 (optional) check_type: [value]
 lockout_policy: [LOCKOUT_POLICY_TYPE]
</custom_item>
```

Cet élément de stratégie contrôle les valeurs définies dans « Paramètres de sécurité -> Stratégies de comptes -> Stratégie de verrouillage du compte ».

Le contrôle est effectué en invoquant la fonction **`NetUserModalsGet`** avec le niveau 3.

Cet élément utilise le champ **`lockout_policy`** pour décrire l'élément de la stratégie de mot de passe qui doit être vérifié. Les types autorisés sont :

- LOCKOUT_DURATION ("Account lockout duration"- Durée de verrouillage de comptes)
  `value_type: TIME_MINUTE`
  `value_data: DWORD ou RANGE [durée en minutes]`

- LOCKOUT_THRESHOLD ("Account lockout threshold"- Seuil de verrouillage de comptes)
  `value_type: POLICY_DWORD`
  `value_data: DWORD ou RANGE [durée en jours]`

- LOCKOUT_RESET ("Reset lockout account counter after"- Réinitialiser le compteur de verrouillages du compte après)
  `value_type: TIME_MINUTE`
  `value_data: DWORD ou RANGE [durée en minutes]`

Voici un exemple :

```
 <custom_item>
  type: LOCKOUT_POLICY
  description: "Reset lockout account counter after"
  value_type: TIME_MINUTE
  value_data: 120
  lockout_policy: LOCKOUT_RESET
 </custom_item>
```

KERBEROS_POLICY

Utilisation

```
<custom_item>
 type: KERBEROS_POLICY
 description: ["description"]
 value_type: [VALUE_TYPE]
 value_data: [value]
 (optional) check_type: [value]
 kerberos_policy: [KERBEROS_POLICY_TYPE]
</custom_item>
```

Cet élément de stratégie contrôle les valeurs définies dans « Paramètres de sécurité -> Stratégies de comptes -> Stratégie Kerberos ».

Le contrôle est effectué en invoquant la fonction **`NetUserModalsGet`** avec le niveau 1.

Cet élément utilise le champ **`kerberos_policy`** pour décrire l'élément de la stratégie de mot de passe qui doit être vérifié. Les types autorisés sont :

- USER_LOGON_RESTRICTIONS ("Enforce user logon restrictions" - Appliquer les restrictions pour l'ouverture de session)
  `value_type: POLICY_SET`
  `value_data: "Enabled" ou "Disabled"`

- SERVICE_TICKET_LIFETIME ("Maximum lifetime for service ticket" - Durée de vie maximale du ticket de service)
  `value_type: TIME_MINUTE`
  `value_data: DWORD ou RANGE [durée en minutes]`

- USER_TICKET_LIFETIME ("Maximum lifetime for user ticket" - Durée de vie maximale du ticket d'utilisateur)
  `value_type: TIME_HOUR`
  `value_data: DWORD ou RANGE [durée en heures]`

- USER_TICKET_RENEWAL_LIFETIME ("Maximum lifetime for user renewal ticket" - Durée de vie maximale pour le renouvellement du ticket utilisateur)
  `value_type: TIME_DAY`
  `value_data: DWORD ou RANGE [durée en jours]`

- CLOCK_SYNCHRONIZATION_TOLERANCE ("Maximum tolerance for computer clock synchronization" - Tolérance maximale pour la synchronisation de l'horloge de l'ordinateur)
  `value_type: TIME_MINUTE`
  `value_data: DWORD ou RANGE [durée en minutes]`

La stratégie Kerberos peut seulement être vérifiée par rapport à un KDC (Key Distribution Center, centre de distribution de clés), qui est normalement un contrôleur de domaine sous Windows.

Exemple :

```
 <custom_item>
  type: KERBEROS_POLICY
  description: "Maximum lifetime for user renewal ticket"
  value_type: TIME_DAY
  value_data: 12
  kerberos_policy: USER_TICKET_RENEWAL_LIFETIME
 </custom_item>
```

AUDIT_POLICY

Utilisation

```
<custom_item>
 type: AUDIT_POLICY
 description: ["description"]
 value_type: [VALUE_TYPE]
 value_data: [value]
 (optional) check_type: [value]
 audit_policy: [PASSWORD_POLICY_TYPE]
</custom_item>
```

Cet élément de stratégie contrôle les valeurs définies dans « Paramètres de sécurité -> Stratégies locales -> Stratégie d'audit ».

Le contrôle est effectué en invoquant la fonction **`LsaQueryInformationPolicy`** avec le niveau **`PolicyAuditEventsInformation`**.

Cet élément utilise le champ **`audit_policy`** pour décrire l'élément de la stratégie de mot de passe qui doit être vérifié. Les types autorisés sont :

- AUDIT_ACCOUNT_LOGON ("Audit account logon events" - Auditer les événements de connexion aux comptes)
- AUDIT_ACCOUNT_MANAGER ("Audit account management" - Auditer la gestion des comptes)
- AUDIT_DIRECTORY_SERVICE_ACCESS ("Audit directory service access" - Auditer l'accès au service d'annuaire)
- AUDIT_LOGON ("Audit logon events" - Auditer les événements de connexion)
- AUDIT_OBJECT_ACCESS ("Audit object access" - Auditer l'accès aux objets)
- AUDIT_POLICY_CHANGE ("Audit policy change" - Auditer les modifications de stratégie)
- AUDIT_PRIVILEGE_USE ("Audit privilege use" - Auditer l'utilisation des privilèges)
- AUDIT_DETAILED_TRACKING ("Audit process tracking" - Auditer le suivi des processus)
- AUDIT_SYSTEM ("Audit system events" - Auditer les événements système)

```
value_type: AUDIT_SET
value_data: "No auditing", "Success", "Failure", "Success, Failure"
```

Un espace est requis dans « Success, Failure » (Réussite, Échec).

Exemple :

```
 <custom_item>
  type: AUDIT_POLICY
  description: "Audit policy change"
  value_type: AUDIT_SET
  value_data: "Failure"
  audit_policy: AUDIT_POLICY_CHANGE
 </custom_item>
```

AUDIT_POLICY_SUBCATEGORY

Utilisation

```
<custom_item>
 type: AUDIT_POLICY_SUBCATEGORY
 description: ["description"]
 value_type: [VALUE_TYPE]
 value_data: [value]
 (optional) check type: [value]
 audit_policy_subcategory: [SUBCATEGORY_POLICY_TYPE]
</custom_item>
```

Cet élément de stratégie contrôle les valeurs répertoriées dans auditpol /get /category:*.

Le contrôle est effectué en exécutant cmd.exe auditpol /get /category:* via WMI.

Cet élément utilise le champ **`audit_policy_subcategory`** pour déterminer la sous-catégorie qui doit être vérifiée. Le ou les SUBCATEGORY_POLICY_TYPE (types de sous-catégorie de stratégie) sont :

- Security State Change (Modification de l'état de la sécurité)
- Security System Extension (Extension système de sécurité)
- System Integrity (Intégrité du système)
- IPsec Driver (Pilote IPsec)
- Other System Events (Autres événements système)
- Logon (Ouverture de session)
- Logoff (Fermeture de session)
- Account Lockout (Verrouillage de compte)
- IPsec Main Mode (Mode principal IPsec)
- IPsec Quick Mode (Mode rapide IPsec)
- IPsec Extended Mode (Mode étendu IPsec)
- Special Logon (Ouverture de session spéciale)
- Other Logon/Logoff Events (Autres événements d'ouverture/fermeture de session)
- Network Policy Server [Serveur NPS (Network Policy Server)]
- File System (Système de fichiers)
- Registry (Registre)

- Kernel Object (Objet de noyau)
- SAM
- Certification Services (Services de certification)
- Application Generated (Généré par application)
- Handle Manipulation (Manipulation de handle)
- File Share (Partage de fichiers)
- Filtering Platform Packet Drop (Rejet de paquet par la plateforme de filtrage)
- Filtering Platform Connection (Connexion de la plateforme de filtrage)
- Other Object Access Events (Autres événements d'accès à l'objet)
- Sensitive Privilege Use (Utilisation de privilèges sensibles)
- Non Sensitive Privilege Use (Utilisation de privilèges non sensibles)
- Other Privilege Use Events (Autres événements d'utilisation de privilèges)
- Process Creation (Création du processus)
- Process Termination (Fin du processus)
- DPAPI Activity (Activité DPAPI)
- RPC Events (Événements RPC)
- Audit Policy Change (Auditer les modifications de stratégie)
- Authentication Policy Change (Modification de la stratégie d'authentification)
- Authorization Policy Change (Modification de la stratégie d'autorisation)
- MPSSVC Rule-Level Policy Change (Modification de la stratégie de niveau règle MPSSVC)
- Filtering Platform Policy Change (Modification de la stratégie de plateforme de filtrage)
- Other Policy Change Events (Autres événements de modification de stratégie)
- User Account Management (Gestion des comptes d'utilisateur)
- Computer Account Management (Gestion des comptes d'ordinateur)
- Security Group Management (Gestion des groupes de sécurité)
- Distribution Group Management (Gestion des groupes de distribution)
- Application Group Management (Gestion des groupes d'applications)
- Other Account Management Events (Autres événements de gestion des comptes)
- Directory Service Access (Accès au service d'annuaire)
- Directory Service Changes (Modification du service d'annuaire)
- Directory Service Replication (Réplication du service d'annuaire)
- Detailed Directory Service Replication (Réplication du service d'annuaire détaillé)
- Credential Validation (Validation des informations d'identification)
- Kerberos Service Ticket Operations (Opérations de ticket du service Kerberos)
- Other Account Logon Events (Autres événements d'ouverture de session)

```
value_type: AUDIT_SET
value_data: "No auditing", "Success", "Failure", "Success, Failure"
```

Un espace est requis dans « Success, Failure » (Réussite, Échec).

Ce contrôle concerne uniquement les systèmes Windows Vista/2008 Server et plus récents. Si un pare-feu est activé, en plus d'ajouter WMI comme exception dans les paramètres du pare-feu, vous devez activer "Windows Firewall : Allow inbound remote administration exception" (Pare-feu Windows : autoriser l'exception d'administration à distance entrante) dans les paramètres du pare-feu en utilisant **`gpedit.msc`**. Ce contrôle pourrait ne pas fonctionner sur les systèmes Vista/2008 qui ne sont pas en langue anglaise ou les systèmes sur lesquels auditpol n'est pas installé.

Exemple :

```
<custom_item>
 type: AUDIT_POLICY_SUBCATEGORY
 description: "AUDIT Security State Change"
```

```
  value_type: AUDIT_SET
  value_data: "success, failure"
  audit_policy_subcategory: "Security State Change"
 </custom_item>
```

AUDIT_POWERSHELL

Utilisation

```
<custom_item>
 type: AUDIT_POWERSHELL
 description: "Powershell check"
 value_type: [value_type]
 value_data: [value]
 powershell_args: ["arguments for powershell.exe"]
 (optional) only_show_cmd_output: YES or NO
 (optional) check_type: [CHECK_TYPE]
 (optional) severity: ["HIGH" or "MEDIUM" or "LOW"]
 (optional) powershell_option: CAN_BE_NULL
 (optional) powershell_console_file: "C:\Program Files\Microsoft\Exchange
Server\ExShell.psc1"
</custom_item>
```

Ce contrôle exécute **`powershell.exe`** sur le serveur distant de même que les arguments fournis avec « **`powershell_args`** » et renvoie la sortie de la commande si « **`only_show_cmd_output`** » est configuré sur YES ou compare le résultat à « **`value_data`** » si **`value_data`** est spécifié.

Types associés :

Cet élément utilise le champ « **`powershell_args`** » afin de spécifier les arguments qui doivent être fournis à **`powershell.exe`**. Si powershell.exe ne se trouve pas à l'emplacement par défaut, vous devez utiliser le mot clé **`powershell_console_file`** pour spécifier l'emplacement. Seuls les cmdlets « `get-` » sont pris en charge actuellement. Par exemple :

- `get-hotfix | where-object {$_.hotfixid -ne 'File 1'} | select Description,HotFixID,InstalledBy | format-list`
- `get-wmiobject win32_service | select caption,name, state| format-list`
- `(get-WmiObject -namespace root\MicrosoftIISv2 -Class IIsWebService).ListWebServiceExtensions().Extensions`
- `get-wmiobject -namespace root\cimv2 -class win32_product | select Vendor,Name,Version | format-list`
- `get-wmiobject -namespace root\cimv2\power -class Win32_powerplan | select description,isactive | format-list`

L'élément utilise le champ facultatif « **`only_show_cmd_output`** » s'il convient de signaler tous les résultats de la commande :

- `only_show_cmd_output: YES` ou `NO`

Autres considérations :

1. Si vous définissez « **only_show_cmd_output** » et que vous souhaitez configurer le niveau de gravité de la sortie, vous pouvez utiliser la balise severity pour modifier la gravité. La valeur par défaut est INFO.
2. Powershell n'est pas installé par défaut sur certains systèmes d'exploitation Windows (par exemple, XP, 2003) et ce contrôle ne renverra aucun résultat sur ces systèmes. Vérifiez par conséquent que Powershell est installé sur la cible distante avant d'utiliser ce contrôle.
3. Pour que ce contrôle fonctionne correctement, le service WMI doit être activé. Par ailleurs, configurez le pare-feu sur « Allow inbound remote administration exception » (Autoriser l'exception d'administration à distance entrante).
4. Les alias cmdlet (par exemple, « `gps` » au lieu de « `Get-Process` ») ne sont pas autorisés.

Exemple :

Cet exemple exécute le cmdlet Powershell « Get-Hotfix », spécifie un objet where afin de ne pas sélectionner les hotfix dotés de l'id « File 1 », puis renvoie les informations Description, HotfixID, Installedby sous forme de liste.

```
<custom_item>
 type: AUDIT_POWERSHELL
 description: "Show Installed Hotfix"
 value_type: POLICY_TEXT
 value_data: ""
 powershell_args: "get-hotfix | where-object {$_.hotfixid -ne 'File 1'} | select
       Description,HotFixID,InstalledBy | format-list"
 only_show_cmd_output: YES
</custom_item>
```

Exemple :

Cet exemple vérifie si le service Windows « WinRM » est en cours d'exécution.

```
<custom_item>
 type: AUDIT_POWERSHELL
 description: "Check if WinRM service is running"
 value_type: POLICY_TEXT
 value_data: "Running"
 powershell_args: "get-wmiobject win32_service | where-object {$_.name -eq 'WinRM' -
       and $_.state -eq 'Running'} | select state"
 check_type: CHECK_REGEX
</custom_item>
```

AUDIT_FILEHASH_POWERSHELL

Utilisation

```
<custom_item>
 type: AUDIT_FILEHASH_POWERSHELL
 description: "Powershell FileHash Check"
 value_type: POLICY_TEXT
 file: "[FILE]"
 value_data: "[FILE HASH]"
</custom_item>
```

Ce contrôle exécute **`powershell.exe`** sur le serveur distant avec les informations fournies afin de comparer une empreinte numérique (valeur de hachage) de fichier attendue à l'empreinte numérique de fichier sur le système.

Autres considérations :

- Par défaut, une empreinte numérique MD5 du fichier fait l'objet d'une comparaison, mais les utilisateurs peuvent comparer les empreintes numériques générées à l'aide de l'algorithme SHA1, SHA256, SHA384, SHA512 ou RIPEMD160.
- Pour garantir le bon fonctionnement de la vérification, PowerShell doit être installé et WMI doit être activé sur la cible.

Exemple :

Cet exemple compare une empreinte numérique MD5 fournie à l'empreinte numérique de fichier pour `C:\test\test2.zip`.

```
<custom_item>
 type: AUDIT FILEHASH POWERSHELL
 description: "Audit FILEHASH - MD5"
 value_type: POLICY_TEXT
 file: "C:\test\test2.zip"
 value_data: "8E653F7040AC4EA8E315E838CEA83A04"
</custom_item>
```

Exemple :

Cet exemple compare une empreinte numérique SHA1 fournie à l'empreinte numérique de fichier pour `C:\test\test3.zip`.

```
<custom_item>
 type: AUDIT_FILEHASH_POWERSHELL
 description: "Audit FILEHASH - SHA1"
 value_type: POLICY_TEXT
 file: "C:\test\test3.zip"
 value_data: "0C4B0AF91F62ECCED3B16D35DE50F66746D6F48F"
 hash_algorithm: SHA1
</custom_item>
```

AUDIT_IIS_APPCMD

Utilisation

```
<custom_item>
 type: AUDIT_IIS_APPCMD
 description: "Test appcmd output"
 value_type: [value_type]
 value_data: [value]
 appcmd_args: ["arguments for appcmd.exe"]
 (optional) only_show_cmd_output: YES or NO
 (optional) check_type: [CHECK_TYPE]
 (optional) severity: ["HIGH" or "MEDIUM" or "LOW"]
</custom_item>
```

Ce contrôle exécute **appcmd.exe** sur un serveur exécutant IIS, ainsi que les arguments spécifiés avec « **appcmd_args** », et il détermine la conformité en comparant la sortie à **value_data**. Dans certains cas (par exemple, configuration de liste), il peut être souhaitable de signaler uniquement les résultats de la commande. Il convient d'utiliser « **only_show_cmd_output** » dans ces cas spécifiques.

Ce contrôle s'applique uniquement à Internet Information Services (IIS) version 7 sous Windows.

Cet élément utilise le champ « **appcmd_args** » afin de spécifier les arguments qui doivent être fournis à **appcmd.exe**. Seules les commandes « list » peuvent actuellement être spécifiées.

- `list sites`
- `list AppPools /processModel.identityType:ApplicationPoolIdentity`
- `list config`
- `list config -section:system.web/authentication`
- `list app`

L'élément utilise le champ facultatif « **only_show_cmd_output** » s'il convient de signaler tous les résultats de la commande.

Exemple :

Ce contrôle compare le résultat de « `appcmd.exe list AppPools /processModel.identityType:ApplicationPoolIdentity` » à **value_data**, et indique une réussite uniquement si la sortie contient 'APPPOOL "DefaultAppPool"'.

```
 <custom_item>
  type: AUDIT_IIS_APPCMD
  description: "Set Default Application Pool Identity to Least Privilege Principal"
  value_type: POLICY_TEXT
  value_data: 'APPPOOL "DefaultAppPool"'
  appcmd_args: "list AppPools /processModel.identityType:ApplicationPoolIdentity"
  check_type: CHECK_REGEX
 </custom_item>
```

AUDIT_ALLOWED_OPEN_PORTS

Utilisation

```
<custom_item>
 type: AUDIT_ALLOWED_OPEN_PORTS
 description: "Audit Open Ports"
 value_type: [value_type]
 value_data: [value]
 port_type: [port_type]
<item>
```

Ce contrôle interroge la liste des ports TCP/UDP ouvertes sur la cible et les compare à une liste autorisée des ports. Le contrôle fait appel à la sortie de « netstat –ano » ou de « netstat –an » pour obtenir une liste des ports ouverts, puis il confirme que les ports sont bien ouverts en vérifiant leur état à l'aide de la commande (get_port_state()/get_udp_port_state()).

Considérations :

- **value_data** accepte également une expression rationnelle (regex) comme plage de ports ; une expression du type 8[0-9]+ est donc également possible.

Exemples :

L'exemple suivant compare « **value_data** » à une liste des ports TCP ouverts sur la cible :

```
<custom_item>
 type: AUDIT_ALLOWED_OPEN_PORTS
 description: "Audit TCP OPEN PORTS"
 value_type: POLICY_PORTS
 value_data: "80,135,445,902,912,1024,1025,3389,5900,8[0-
      9]+,18208,32111,38311,47001,139"
 port_type: TCP
</custom_item>
```

L'exemple suivant compare « **value_data** » à une liste des ports UDP ouverts sur la cible :

```
<custom_item>
 type: AUDIT_ALLOWED_OPEN_PORTS
 description: "Audit UDP OPEN PORTS"
 value_type: POLICY_PORTS
 value_data: "161,445,500,1026,4501,123,137,138,5353"
 port_type: UDP
</custom_item>
```

AUDIT_DENIED_OPEN_PORTS

Utilisation

```
<custom_item>
 type: AUDIT_DENIED_OPEN_PORTS
 description: "Audit Denied Open Ports"
 value_type: [value_type]
 value_data: [value]
 port_type: [port_type]
<item>
```

Ce contrôle interroge la liste des ports TCP/UDP ouverts sur la cible et les compare à une liste des ports refusés. Le contrôle fait appel à la sortie de « netstat –ano » ou de « netstat –an » pour obtenir une liste des ports ouverts, puis il confirme que les ports sont bien ouverts en vérifiant leur état à l'aide de la commande (get_port_state()/get_udp_port_state()).

Les types autorisés sont :

- `value_type: POLICY_PORTS`
- `value_data: "80,135,445,902,912,1024,1025,3389,5900,8[0-9]+,18208,32111,38311,47001,139"`
- `port_type: TCP ou UDP`

Considérations :

- **value_data** accepte également une expression rationnelle (regex) comme plage de ports ; une expression du type 8[0-9]+ est donc également possible.

Exemples :

L'exemple suivant compare « **value_data** » à une liste des ports TCP ouverts sur la cible.

```
<custom_item>
 type: AUDIT_DENIED_OPEN_PORTS
 description: "Audit TCP OPEN PORTS"
 value_type: POLICY_PORTS
 value_data: "80,443"
 port_type: TCP
</custom_item>
```

L'exemple suivant compare « **value_data** » à une liste des ports UDP ouverts sur la cible.

```
<custom_item>
 type: AUDIT_DENIED_OPEN_PORTS
 description: "Audit UDP OPEN PORTS"
 value_type: POLICY_PORTS
 value_data: "161,5353"
 port_type: UDP
</custom_item>
```

AUDIT_PROCESS_ON_PORT

Utilisation

```
<custom_item>
 type: AUDIT_PROCESS_ON_PORT
 description: "Audit Process on Port"
 value_type: [value_type]
 value_data: [value]
 port_type: [port_type]
 port_no: [port_no]
 port_option: [port_option]
 check_type: CHECK_TYPE
<item>
```

Ce contrôle interroge le processus en cours d'exécution sur un port donné. Le contrôle fait appel à la sortie de « netstat -ano » ou de « tasklist /svc » pour déterminer le processus qui est en cours d'exécution sur un port TCP/UDP donné.

Les types autorisés sont :

- `value_type: POLICY_TEXT`
- `value_data:` Chaîne arbitraire, par exemple, `« foo.exe »`
- `port_type: TCP` ou `UDP`
- `port_no:` numéro de port, par exemple, 80, 445

- `port_option: CAN_BE_CLOSED`

Considérations :

- Lorsque **`port_option`** est paramétré sur CAN_BE_CLOSED, le contrôle renvoie un résultat indiquant une réussite (PASS) si le port n'est pas ouvert sur le système distant ; sinon, il génère une erreur.
- Windows 2000 et les versions antérieures ne prennent pas en charge le contrôle « netstat –ano », qui fonctionne uniquement sur Windows XP et les versions ultérieures.

Exemples :

L'exemple suivant vérifie si le processus en cours d'exécution sur le port tcp 5900 est « vss.exe » ou « vssrvc.exe ».

```
<custom_item>
 type: AUDIT_PROCESS_ON_PORT
 description: "Audit OPEN PORT SERVICE"
 value_type: POLICY_TEXT
 value_data: "vssrvc.exe" || "vss.exe"
 port_type: TCP
 port_no: "5900"
 port_option: CAN_BE_CLOSED
</custom_item>
```

L'exemple suivant est similaire au premier, sauf qu'il démontre l'utilisation de check_type.

```
<custom_item>
 type: AUDIT_PROCESS_ON_PORT
 description: "Audit Process on Port - check_regex"
 value_type: POLICY_TEXT
 value_data: "foo.exe" || "vss.+"
 port_type: TCP
 port_no: "5900"
 check_type: CHECK_REGEX
</custom_item>
```

CHECK_ACCOUNT

Utilisation

```
<custom_item>
 type: CHECK_ACCOUNT
 description: ["description"]
 value_type: [VALUE_TYPE]
 value_data: [value]
 account_type: [ACCOUNT_TYPE]
 (optional) check_type: [CHECK_TYPE]
</custom_item>
```

Cet élément de stratégie contrôle les valeurs suivantes définies dans « Paramètres de sécurité -> Stratégies locales -> Options de sécurité ».

- Accounts: Administrator account status (Comptes : état de compte d'administrateur)

- Accounts: Guest account status (Comptes : état de compte d'invité)
- Accounts: Rename administrator account (Comptes : renommer le compte administrateur)
- Accounts: Rename guest account (Comptes : renommer le compte invité)

Le contrôle est effectué en invoquant la fonction **`LsaQueryInformationPolicy`** avec le niveau **`PolicyAccountDomainInformation`** pour obtenir le SID de domaine/système, **`LsaLookupSid`** pour obtenir les noms d'administrateur et d'invité et **`NetUserGetInfo`** pour obtenir les informations de compte.

Cet élément utilise le champ **`account_type`** pour décrire le compte qui doit être vérifié. Les types autorisés sont :

- ADMINISTRATOR_ACCOUNT (“Accounts: Administrator account status” [Comptes : état de compte d'administrateur])
  `value_type: POLICY_SET`
  `value_data: "Enabled" ou "Disabled"`
- GUEST_ACCOUNT (“Accounts: Guest account status” [Comptes : état de compte d'invité])
  `value_type: POLICY_SET`
  `value_data: "Enabled" ou "Disabled"`
- ADMINISTRATOR_ACCOUNT (“Accounts: Rename administrator account” [Comptes : renommer le compte administrateur])
  `value_type: POLICY_TEXT`
  `value_data: "TEXT HERE" [nom de l'administrateur]`
  `check_type: [CHECK_TYPE] (l'une des valeurs check_type possibles)`
- GUEST_ACCOUNT (“Accounts: Rename guest account” [Comptes : renommer le compte invité])
  `value_type: POLICY_TEXT`
  `value_data: "TEXT HERE" [nom de l'invité]`
  `check_type: [CHECK_TYPE] (l'une des valeurs check_type possibles)`

Suivant la partie d'authentification de domaine, les comptes de système local ou les comptes de domaine peuvent être contrôlés.

Exemples :

```
<custom_item>
 type: CHECK_ACCOUNT
 description: "Accounts: Guest account status"
 value_type: POLICY_SET
 value_data: "Disabled"
 account_type: GUEST_ACCOUNT
</custom_item>

<custom_item>
 type: CHECK_ACCOUNT
 description: "Accounts: Rename administrator account"
 value_type: POLICY_TEXT
 value_data: "Dom_adm"
 account_type: ADMINISTRATOR_ACCOUNT
</custom_item>

<custom_item>
 type: CHECK_ACCOUNT
```

```
  description: "Accounts: Rename administrator account"
  value_type: POLICY_TEXT
  value_data: "Administrator"
  account_type: ADMINISTRATOR_ACCOUNT
  check_type: CHECK_NOT_EQUAL
 </custom_item>
```

CHECK_LOCAL_GROUP

Utilisation

```
<custom_item>
 type: CHECK_LOCAL_GROUP
 description: ["description"]
 value_type: [VALUE_TYPE]
 value_data: [value]
 group_type: [GROUP_TYPE]
 (optional) check_type: [CHECK_TYPE]
</custom_item>
```

Cet élément de stratégie contrôle les noms de groupe et l'état des groupes répertoriés dans **`lusmgr.msc`**.

Cet élément utilise le champ **`group_type`** pour décrire le compte qui doit être vérifié. Les types autorisés sont :

- ADMINISTRATORS_GROUP
- USERS_GROUP
- GUESTS_GROUP
- POWER_USERS_GROUP
- ACCOUNT_OPERATORS_GROUP
- SERVER_OPERATORS_GROUP
- PRINT_OPERATORS_GROUP
- BACKUP_OPERATORS_GROUP
- REPLICATORS_GROUP

Les types autorisés pour le champ **`value_type`** sont :

- POLICY_SET (l'état du groupe est contrôlé)
  `value_type: POLICY_SET`
  `value_data: "Enabled" ou "Disabled"`
- POLICY_TEXT (le nom du groupe est contrôlé)
  `value_type: POLICY_TEXT`
  `value_data: "Guests1"` (Dans ce cas, **`value_data`** peut être une chaîne de texte quelconque)

Exemples :

```
<custom_item>
 type: CHECK_LOCAL_GROUP
 description: "Local Guest group must be enabled"
 value_type: POLICY_SET
 value_data: "enabled"
 group_type: GUESTS_GROUP
 check_type: CHECK_EQUAL
</custom_item>
```

```
<custom_item>
 type: CHECK_LOCAL_GROUP
 description: "Guests group account name should be Guests"
 value_type: POLICY_TEXT
 value_data: "Guests"
 group_type: GUESTS_GROUP
 check_type: CHECK_EQUAL
</custom_item>
```

```
<custom_item>
 type: CHECK_LOCAL_GROUP
 description: "Guests group account name should not be Guests"
 value_type: POLICY_TEXT
 value_data: "Guests"
 group_type: GUESTS_GROUP
 check_type: CHECK_NOT_EQUAL
</custom_item>
```

ANONYMOUS_SID_SETTING

Utilisation

```
<custom_item>
 type: ANONYMOUS_SID_SETTING
 description: ["description"]
 value_type: [VALUE_TYPE]
 value_data: [value]
 (optional) check_type: [value]
</custom_item>
```

Cet élément de stratégie contrôle les valeurs suivantes définies dans « Paramètres de sécurité -> Stratégies locales -> Options de sécurité -> Accès réseau : Permet la traduction de noms/SID anonymes ». Le contrôle est effectué en invoquant la fonction **`LsaQuerySecurityObject`** sur l'identificateur de stratégie LSA.

Les types autorisés sont :

```
value_type: POLICY_SET
value_data: "Enabled" ou "Disabled"
```

Lorsque cet audit est utilisé, veuillez noter que cette stratégie :

- est un contrôle de permission sur le service LSA

- vérifie si ANONYMOUS_USER a défini l'indicateur POLICY_LOOKUP_NAMES
- est déprécié sur Windows 2003 parce qu'un utilisateur anonyme ne peut pas accéder au pipe LSA

Exemple :

```
<custom_item>
 type: ANONYMOUS_SID_SETTING
 description: "Network access: Allow anonymous SID/Name translation"
 value_type: POLICY_SET
 value_data: "Disabled"
</custom_item>
```

SERVICE_POLICY

Le bon fonctionnement de ce contrôle nécessite que le système Windows distant ait accès au registre distant.

Utilisation

```
<custom_item>
 type: SERVICE_POLICY
 description: ["description"]
 value_type: [VALUE_TYPE]
 value_data: [value]
 (optional) check type: [value]
 service_name: ["service name"]
</custom_item>
```

Cet élément de stratégie vérifie les valeurs de démarrage définies dans « System Services » (Services système). Le contrôle est effectué en invoquant la fonction **`RegQueryValueEx`** sur les clés suivantes :

- key: "SYSTEM\CurrentControlSet\Services\" + service_name
- item: "Start"

Les types autorisés sont :

```
value_type: SERVICE_SET
value_data: "Automatic", "Manual" ou "Disabled"
svc_option: CAN_BE_NULL ou CAN_NOT_BE_NULL
```

Le champ **`service_name`** correspond au nom RÉEL du service. Ce nom peut être obtenu en :

1. lançant le Panneau de configuration Services (dans Outils d'administration)
2. sélectionnant le service souhaité
3. ouvrant la boîte de dialogue des propriétés (cliquez avec le bouton droit -> Propriétés)
4. extrayant la portion « Service name » (Nom du service)

Le paramètre des autorisations de service peut être contrôlé avec un élément SERVICE_PERMISSIONS.

Exemple :

```
<custom_item>
 type: SERVICE_POLICY
 description: "Background Intelligent Transfer Service"
 value_type: SERVICE_SET
 value_data: "Disabled"
 service_name: "BITS"
</custom_item>
```

GROUP_MEMBERS_POLICY

Utilisation

```
<custom_item>
 type: GROUP_MEMBERS_POLICY
 description: ["description"]
 value_type: [value type]
 value_data: [value]
 (optional) check_type: [value]
 group_name: ["group name"]
</custom_item>
```

Cet élément de stratégie vérifie qu'une liste spécifique d'utilisateurs est présente dans un ou plusieurs groupes.

Le type autorisé est :

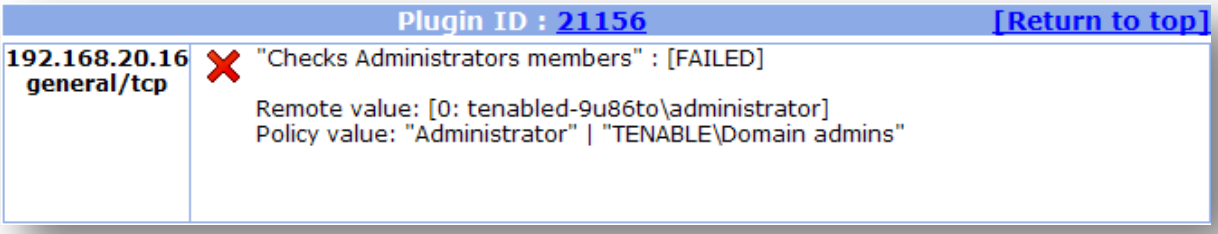
```
value_type: POLICY_TEXT ou POLICY_MULTI_TEXT
value_data: "user1" && "user2" && ... && "usern"
```

Lorsque cet audit est utilisé, un nom d'utilisateur peut être spécifié avec un nom de domaine, tel que « MONDOMAINE\Jean Dupont », et le champ **group_name** spécifie un seul groupe à vérifier.

Un fichier Nessus **.audit** simple peut spécifier plusieurs éléments de clients différents, et il est donc très facile de vérifier les listes d'utilisateurs dans plusieurs groupes. Voici un exemple de stratégie **.audit** qui recherche le groupe « Administrators » (Administrateurs) contenant seulement l'utilisateur « Administrator » et l'utilisateur « TENABLE\Domain admins » :

```
<custom_item>
 type: GROUP_MEMBERS_POLICY
 description: "Checks Administrators members"
 value_type: POLICY_MULTI_TEXT
 value_data: "Administrator" && "TENABLE\Domain admins"
 group_name: "Administrators"
</custom_item>
```

Voici un exemple de capture d'écran de l'exécution du contenu du fichier **.audit** ci-dessus avec un serveur Windows 2003 :



USER_GROUPS_POLICY

Utilisation

```
<custom_item>
 type: USER_GROUPS_POLICY
 description: ["description"]
 value_type: [value type]
 value_data: [value]
 (optional) check_type: [value]
 user_name: ["user name"]
</custom_item>
```

Cet élément de stratégie vérifie qu'un utilisateur Windows appartient aux groupes spécifiés dans **`value_data`**. Lorsque cet audit est utilisé, vous pouvez seulement tester les utilisateurs de domaine en fonction d'un contrôleur de domaine. Cette vérification ne concerne pas les utilisateurs intégrés tels que le « Local Service » (Service local).

Exemple :

```
 <custom_item>
  type: USER_GROUPS_
  description: "3.72 DG0005: DBMS administration OS accounts"
  info: "Checking that the 'dba' account is a member of required groups only."
  info: "Modify the account/groups in this audit to match your environment."
  value_type: POLICY_MULTI_TEXT
  value_data: "Users" && "SQL Server DBA" && "SQL Server Users"
  user_name: "dba"
 </custom_item>
```

USER_RIGHTS_POLICY

Utilisation

```
<custom_item>
 type: USER_RIGHTS_POLICY
 description: ["description"]
 value_type: [value type]
 value_data: [value]
 (optional) check_type: [value]
 right_type: [right]
</custom_item>
```

Cet élément de stratégie contrôle la valeur suivante définie dans « Paramètres de sécurité -> Stratégies locales -> Attribution des droits utilisateur ». Le contrôle est effectué en invoquant la fonction **`LsaEnumerateAccountsWithUserRight`** sur l'identificateur de stratégie LSA.

Le champ **`right_type`** correspond au droit de tester. Les valeurs autorisées sont :

```
right_type: RIGHT
```

Où **`RIGHT`** peut être :

```
SeAssignPrimaryTokenPrivilege
SeAuditPrivilege
SeBackupPrivilege
SeBatchLogonRight
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeCreatePagefilePrivilege
SeCreatePermanentPrivilege
SeCreateTokenPrivilege
SeDenyBatchLogonRight
SeDenyInteractiveLogonRight
SeDenyNetworkLogonRight
SeDenyRemoteInteractiveLogonRight
SeDenyServiceLogonRight
SeDebugPrivilege
SeEnableDelegationPrivilege
SeImpersonatePrivilege
SeIncreaseBasePriorityPrivilege
SeIncreaseWorkingSetPrivilege
SeIncreaseQuotaPrivilege
SeInteractiveLogonRight
SeLoadDriverPrivilege
SeLockMemoryPrivilege
SeMachineAccountPrivilege
SeManageVolumePrivilege
SeNetworkLogonRight
SeProfileSingleProcessPrivilege
SeRemoteShutdownPrivilege
SeRemoteInteractiveLogonRight
SeReLabelPrivilege
SeRestorePrivilege
SeSecurityPrivilege
SeServiceLogonRight
SeShutdownPrivilege
SeSyncAgentPrivilege
SeSystemEnvironmentPrivilege
SeSystemProfilePrivilege
SeSystemtimePrivilege
SeTakeOwnershipPrivilege
SeTcbPrivilege
SeTimeZonePrivilege
SeUndockPrivilege
SeUnsolicitedInputPrivilege
```

Le type autorisé est

```
value_type: USER_RIGHT
value_data: "user1" && "user2" && "group1" && ... && "groupn"
```

Les tests de droits d'utilisateur effectuent beaucoup de demandes concernant le contrôleur de domaine. Ces tests doivent être inclus dans un fichier de stratégie séparé et lancés seulement contre le contrôleur de domaine et UN système du domaine.

Le type **`right`** ne doit pas être placé entre guillemets car il est analysé comme un jeton.

Exemple :

```
<custom_item>
 type: USER_RIGHTS_POLICY
 description: "Create a token object"
 value_type: USER_RIGHT
 value_data: "Administrators" && "Backup Operators"
 right_type: SeCreateTokenPrivilege
</custom_item>
```

FILE_CHECK

Le bon fonctionnement de ce contrôle nécessite que le système Windows distant ait accès au registre distant.

Utilisation

```
<custom_item>
 type: FILE_CHECK
 description: ["description"]
 value_type: [VALUE_TYPE]
 value data: [value]
 (optional) check_type: [value]
 file_option: [OPTION_TYPE]
</custom_item>
```

Cet élément de stratégie vérifie si le fichier (**`value_data`**) existe ou non (**`file_option`**). Le contrôle est effectué en invoquant la fonction **`CreateFile`**.

Les types autorisés sont :

```
value_type: POLICY_TEXT
value_data: "file name"
file_option: MUST_EXIST ou MUST_NOT_EXIST
```

Exemples :

```
<custom_item>
 type: FILE_CHECK
 description: "Check that win.ini exists in the system root"
 value_type: POLICY_TEXT
 value_data: "%SystemRoot%\win.ini"
 file_option: MUST_EXIST
</custom_item>
```

```
<custom_item>
 type: FILE_CHECK
 description: "Check that bad.exe does not exist in the system root"
 value_type: POLICY_TEXT
 value_data: "%SystemRoot%\bad.exe"
 file_option: MUST_NOT_EXIST
</custom_item>
```

FILE_VERSION

Le bon fonctionnement de ce contrôle nécessite que le système Windows distant ait accès au registre distant.

Utilisation

```
<custom_item>
 type: FILE_VERSION
 description: ["description"]
 value_type: [VALUE_TYPE]
 value_data: [value]
 (optional) check_type: [value]
 file: PATH_TO_FILE
 file_option: [OPTION_TYPE]
 check_type: CHECK_TYPE
</custom_item>
```

Cet élément de stratégie vérifie si la version du fichier spécifiée par le champ **`file`** est supérieure ou égale à la version du fichier à distance par défaut. Le contrôle peut aussi être utilisé pour déterminer si la version du fichier à distance est plus ancienne en utilisant l'option **`check_type`**.

Les types autorisés sont :

```
value_type: POLICY_FILE_VERSION
value_data: "file version"
file_option: MUST_EXIST ou MUST_NOT_EXIST
```

Exemples :

```
<custom_item>
 type: FILE_VERSION
 description: "Audit for C:\WINDOWS\SYSTEM32\calc.exe"
 value_type: POLICY_FILE_VERSION
 value_data: "1.1.1.1"
 file: "C:\WINDOWS\SYSTEM32\calc.exe"
</custom_item>
```

```
<custom_item>
 type: FILE_VERSION
 description: "Audit for C:\WINDOWS\SYSTEM32\calc.exe"
 value_type: POLICY_FILE_VERSION
```

```
  value_data: "1.1.1.1"
  file: "C:\WINDOWS\SYSTEM32\calc.exe"
  check_type: CHECK_LESS_THAN
 </custom_item>
```

FILE_PERMISSIONS

Le bon fonctionnement de ce contrôle nécessite que le système Windows distant ait accès au registre distant.

Utilisation

```
<custom_item>
 type: FILE_PERMISSIONS
 description: ["description"]
 value_type: [value_type]
 value_data: [value]
 (optional) check_type: [value]
 file: ["filename"]
 (optional) acl_option: [acl_option]
</custom_item>
```

Cet élément de stratégie vérifie si l'ACL FILE_PERMISSIONS est correcte. Le contrôle est effectué en invoquant la fonction **`GetSecurityInfo`** avec le niveau 7 sur l'identificateur de fichier.

Le type autorisé est :

```
value_type: FILE_ACL
value_data: "ACLname"
file: "PATH\Filename"
```

Les chemins prédéfinis ci-dessous peuvent être utilisés dans le nom de fichier/dossier :

```
%allusersprofile%
%windir%
%systemroot%
%commonfiles%
%programfiles%
%systemdrive%
%systemdirectory%
```

Lorsque cet audit est utilisé, il faut noter les points suivants :

- Le champ **`file`** doit inclure le chemin complet du nom de fichier ou de dossier (par exemple **`C:\WINDOWS\SYSTEM32`**) ou utiliser les mots clés des chemins ci-dessus. Si vous utilisez des mots clés de chemin, le registre distant doit être activé pour permettre à Nessus de déterminer les valeurs variables du chemin.
- Le champ **`value_data`** est le nom d'une ACL définie dans le fichier de la stratégie.
- Le champ **`acl_option`** peut être paramétré sur CAN_BE_NULL (peut être nul) ou CAN_NOT_BE_NULL (ne peut pas être nul) pour forcer une réussite/erreur si le fichier n'existe pas.

Exemples :

```
 <file_acl: "ACL1">

  <user: "Administrators">
   acl_inheritance: "not inherited"
   acl_apply: "This object only"
   acl_allow: "Full Control"
  </user>

  <user: "System">
   acl_inheritance: "not inherited"
   acl_apply: "This object only"
   acl_allow: "Full Control"
  </user>

 </acl>

 <custom_item>
   type: FILE_PERMISSIONS
   description: "Permissions for C:\WINDOWS\SYSTEM32"
   value_type: FILE_ACL
   value_data: "ACL1"
   file: "C:\WINDOWS\SYSTEM32"
 </custom_item>
```

```
 <custom_item>
  type: FILE_PERMISSIONS
  description: "Permissions for C:\WINDOWS\SYSTEM32"
  value_type: FILE_ACL
  value_data: "ACL1"
  file: "%SystemRoot%\SYSTEM32"
 </custom_item>
```

Lorsque le contrôle ci-dessus est exécuté, le module de conformité vérifie si les permissions définies pour « **%SystemRoot%\SYSTEM32** » correspondent à celles décrites dans file_acl ACL1.

FILE_AUDIT

Le bon fonctionnement de ce contrôle nécessite que le système Windows distant ait accès au registre distant.

Utilisation

```
<custom_item>
 type: FILE_AUDIT
 description: ["description"]
 value_type: [value_type]
 value_data: [value]
 (optional) check_type: [value]
 file: ["filename"]
 (optional) acl_option: [acl_option]
</custom_item>
```

Cet élément de stratégie est utilisé pour contrôler les propriétés de vérification (Properties –> Security –> Advanced –> Auditing (Propriétés -> Sécurité -> Avancé -> Audit) d'un fichier ou d'un dossier en utilisant l'ACL spécifiée. Le contrôle est effectué en invoquant la fonction **`GetSecurityInfo`** avec le niveau SACL_SECURITY_INFORMATION sur l'identificateur de fichier.

Le type autorisé est :

```
value_type: FILE_ACL
value_data: "ACLname"
file: "PATH\Filename"
```

Les chemins prédéfinis ci-dessous peuvent être utilisés dans le nom de fichier/dossier :

```
%allusersprofile%
%windir%
%systemroot%
%commonfiles%
%programfiles%
%systemdrive%
%systemdirectory%
```

Lorsque cet audit est utilisé, il faut noter les points suivants :

- Le champ **`file`** doit inclure le chemin complet du nom de fichier ou de dossier (par exemple **`C:\WINDOWS\SYSTEM32`**) ou utiliser les mots clés des chemins ci-dessus. Si vous utilisez des mots clés de chemin, le registre distant doit être activé pour permettre à Nessus de déterminer les valeurs variables du chemin.
- Le champ **`value_data`** est le nom de l'ACL définie dans le fichier de la stratégie.
- Le champ **`acl_option`** peut être paramétré sur CAN_BE_NULL (peut être nul) ou CAN_NOT_BE_NULL (ne peut pas être nul) pour forcer une réussite/ erreur si le fichier n'existe pas.
- Les champs **`acl_allow`** et **`acl_deny`** correspondent aux événements d'audit « Successful » (Réussite) et « Failed » (Échec).

Voici un exemple de fichier **`.audit`** qui applique la fonction FILE_AUDIT, avec un exemple de règle de liste de contrôle d'accès nommée « ACL1 ».

```
<check_type: "Windows" version:"2">
<group_policy: "Audits SYSTEM32 directory for correct auditing permissions">

<file_acl: "ACL1">
 <user: "Everyone">
  acl_inheritance: "not inherited"
  acl_apply: "This folder, subfolders and files"
  acl_deny: "full control"
  acl_allow: "full control"
 </user>
</acl>

<custom_item>
 type: FILE_AUDIT
 description: "Audit for C:\WINDOWS\SYSTEM32"
 value_type: FILE_ACL
 value_data: "ACL1"
 file: "%SystemRoot%\SYSTEM32"
</custom_item>
```

```
</group_policy
</check_type>
```

FILE_CONTENT_CHECK

Le bon fonctionnement de ce contrôle nécessite que le système Windows distant ait accès au registre distant.

Utilisation

```
<custom_item>
 type: FILE_CONTENT_CHECK
 description: ["description"]
 value_type: [value_type]
 value_data: ["filename"]
 (optional) check_type: [value]
 regex: ["regex"]
 expect: ["regex"]
 (optional) file_option: [file_option]
 (optional) avoid_floppy_access
</custom_item>
```

Cet élément de stratégie vérifie si le fichier contient l'expression rationnelle **`regex`** et si cette expression correspond à **`expect`**.

Le contrôle est effectué en invoquant la fonction **`ReadFile`** sur l'identificateur de fichier.

Le fichier est lu via le protocole SMB dans une mémoire tampon sur le serveur Nessus, puis le tampon est traité afin de vérifier la conformité/non-conformité. Les fichiers ne sont pas enregistrés sur le disque du serveur Nessus, ils sont uniquement copiés dans un tampon de mémoire pour analyse.

Le type autorisé est :

```
value_type: POLICY_TEXT
value_data: "PATH\Filename"
regex: "regex"
expect: "regex"
```

Les chemins prédéfinis ci-dessous peuvent être utilisés dans le nom de fichier/dossier :

```
%allusersprofile%
%windir%
%systemroot%
%commonfiles%
%programfiles%
%systemdrive%
```

Lorsque ce type d'audit est utilisé, il faut noter les points suivants :

- Le champ **value_data** doit inclure le chemin complet du nom de fichier ou de dossier (par exemple **C:\WINDOWS\SYSTEM32**) ou utiliser les mots clés des chemins ci-dessus. Si vous utilisez des mots clés de chemin, le registre distant doit être activé pour permettre à Nessus de déterminer les valeurs variables du chemin.
- Le champ **regex** vérifie qu'un élément est présent dans le fichier.
- Le champ **expect** vérifie que l'élément correspond à l'expression rationnelle.
- Le champ **file_option** peut être paramétré sur CAN_BE_NULL (peut être nul) pour forcer une réussite si le fichier n'existe pas.
- Le champ **file_option** peut être paramétré sur CAN_NOT_BE_NULL (ne peut pas être nul) pour forcer une erreur si le fichier existe et est vide.
- Le champ **avoid_floppy_access** peut être configuré pour indiquer à l'audit de ne pas effectuer un contrôle qui se traduirait par l'accès à la disquette. Cette procédure doit être utilisée uniquement si un audit entraîne l'accès à la disquette lorsque l'unité ne contient pas de disque.

Exemple :

```
 <custom_item>
  avoid_floppy_access
  type: FILE_CONTENT_CHECK
  description: "File content for C:\WINDOWS\win.ini"
  value_type: POLICY_TEXT
  value_data: "C:\WINDOWS\win.ini"
  regex: "aif=.*"
  expect: "aif=MPEGVideo"
 </custom_item>
```

FILE_CONTENT_CHECK_NOT

Le bon fonctionnement de ce contrôle nécessite que le système Windows distant ait accès au registre distant.

Utilisation

```
<custom_item>
 type: FILE_CONTENT_CHECK_NOT
 description: ["description"]
 value_type: [value_type]
 value_data: ["filename"]
 (optional) check_type: [value]
 regex: ["regex"]
 expect: ["regex"]
 (optional) file_option: [file_option]
</custom_item>
```

Cet élément de stratégie vérifie si le fichier contient l'expression rationnelle regex et si cette expression ne correspond pas à expect. Le contrôle est effectué en invoquant la fonction **ReadFile** sur l'identificateur de fichier.

Le type autorisé est :

```
value_type: POLICY_TEXT
value_data: "PATH\Filename"
regex: "regex"
expect: "regex"
```

Les chemins prédéfinis ci-dessous peuvent être utilisés dans le nom de fichier/dossier :

```
%allusersprofile%
%windir%
%systemroot%
%commonfiles%
%programfiles%
%systemdrive%
```

Lorsque ce type d'audit est utilisé, il faut noter les points suivants :

- Le champ **value_data** doit inclure le chemin complet du nom de fichier ou de dossier (par exemple **C:\WINDOWS\SYSTEM32**) ou utiliser les mots clés des chemins ci-dessus. Si vous utilisez des mots clés de chemin, le registre distant doit être activé pour permettre à Nessus de déterminer les valeurs variables du chemin.
- Le champ **regex** vérifie qu'un élément est présent dans le fichier.
- Le champ **expect** vérifie que l'élément correspond à l'expression rationnelle.
- Le champ **file_option** peut être paramétré sur CAN_BE_NULL (peut être nul) pour forcer une réussite si le fichier n'existe pas.
- Le champ **file_option** peut être paramétré sur CAN_NOT_BE_NULL (ne peut pas être nul) pour forcer une erreur si le fichier existe et est vide.

Exemple :

```
<custom_item>
 type: FILE_CONTENT_CHECK_NOT
 description: "File content for C:\WINDOWS\win.ini"
 value_type: POLICY_TEXT
 value_data: "C:\WINDOWS\win.ini"
 (optional) check_type: [value]
 regex: "au=.*"
 expect: "au=MPEGVideo2"
 file_option: CAN_NOT_BE_NULL
</custom_item>
```

REG_CHECK

Le bon fonctionnement de ce contrôle nécessite que le système Windows distant ait accès au registre distant.

Utilisation

```
<custom_item>
 type: REG_CHECK
 description: ["description"]
```

```
 value_type: [VALUE_TYPE]
 value_data: [value]
 reg_option: [OPTION_TYPE]
 (optional) check_type: [value]
 (optional) key_item: [item value]
</custom_item>
```

Cet élément de stratégie vérifie si la clé du registre (ou l'élément) existe. Le contrôle est effectué en invoquant les fonctions **RegOpenKeyEx** et **RegQueryValueEx**.

Les types autorisés sont :

```
value_type: POLICY_TEXT
value_data: "key path"
reg_option: MUST_EXIST ou MUST_NOT_EXIST
key_item: "item name"
```

Si le champ **key_item** n'est pas spécifié, cet élément vérifie que le chemin de la clé existe. Sinon, il vérifie que l'élément existe.

Exemple :

```
 <custom_item>
  type: REG_CHECK
  description: "Check the key HKLM\SOFTWARE\Adobe\Acrobat Reader\7.0\AdobeViewer"
  value_type: POLICY_TEXT
  value_data: "HKLM\SOFTWARE\Adobe\Acrobat Reader\7.0\AdobeViewer"
  reg_option: MUST_NOT_EXIST
  key_item: "EULA"
 </custom_item>
```

REGISTRY_SETTING

Le bon fonctionnement de ce contrôle nécessite que le système Windows distant ait accès au registre distant.

Utilisation

```
<custom_item>
 type: REGISTRY_SETTING
 description: ["description"]
 value_type: [VALUE_TYPE]
 value_data: [value]
 reg_key: ["key name"]
 reg_item: ["key item"]
 (optional) check_type: [value]
 (optional) reg_option: [KEY_OPTIONS]
 (optional) reg_enum: ENUM_SUBKEYS
</custom_item>
```

Cet élément de stratégie est utilisé pour contrôler la valeur d'une clé de registre. Beaucoup de contrôles de stratégie dans « Paramètres de sécurité -> Stratégies locales -> Options de sécurité » utilisent cet élément de stratégie. Ce contrôle est effectué en invoquant la fonction **RegQueryValueEx**.

Le champ **reg_key** est le nom de la clé de registre (par exemple « HKLM\SOFTWARE\Microsoft\Driver Signing »). La première partie de la clé (HKLM) est utilisée pour la connexion à la ruche du registre correcte. Le chemin suivant est une désignation statique où le **reg_item** souhaité est localisé.

La ruche HKU (HKEY_USERS) est un cas spécial. Il n'est pas possible de spécifier un SID pour les clés HKU. En fait, le `nbin` itère intérieurement sur chaque SID et réussit seulement si la valeur dans chaque SID est valide.

Par exemple :

```
<custom_item>
 type: REGISTRY_SETTING
 description: "HKU\Control Panel\Desktop\ScreenSaveActive"
 value_type: POLICY_DWORD
 value_data: 1
 reg_key: "HKU\Control Panel\Desktop"
 reg_item: "ScreenSaveActive"
</item>
```

ferait une boucle sur :

```
HKU\S-1-5-18\Control Panel\Desktop\ScreenSaveActive
HKU\S-1-5-19\Control Panel\Desktop\ScreenSaveActive
HKU\S-1-5-20\Control Panel\Desktop\ScreenSaveActive
...
```

et réussirait si l'élément « ScreenSaveActive » est paramétré sur 1 pour tous les SID.

Le champ facultatif **reg_option** peut être paramétré avec CAN_BE_NULL pour forcer la réussite du contrôle si la clé n'existe pas, ou avec la valeur opposée CAN_NOT_BE_NULL.

Une option supplémentaire **reg_enum** avec l'argument « ENUM_SUBKEYS » peut être utilisée pour énumérer une valeur particulière pour toutes les sous-clés d'une clé de registre. Par exemple, beaucoup de logiciels sont répertoriés pour la clé **HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall**. Si vous souhaitez faire correspondre la valeur « CurrentVersion » pour toutes les sous-clés sous « Uninstall » (Désinstaller), utilisez **reg_enum**.

Exemple :

```
<custom_item>
 type: REGISTRY_SETTING
 description: "DBMS network port, protocol, and services (PPS) usage"
 info: "Checking whether TCPDynamicPorts key value is configured (should be blank)."
 value_type: POLICY_TEXT
 value_data: ""
 reg_key: "HKLM\SOFTWARE\Microsoft\Microsoft SQL
       Server\MSSQL.1\MSSQLServer\SuperSocketNetLib\Tcp"
 reg_item: "TCPDynamicPorts"
 reg_enum: ENUM_SUBKEYS
 reg_option: CAN_BE_NULL
</custom_item>
```

Cet audit de la ruche du registre HKU n'inclut pas le SID (identificateur de sécurité) dans le chemin de registre **reg_key**. Dans cet exemple, le **reg_item** spécifié est recherché dans chaque SID HKU.

Exemple :

```
<custom_item>
 type: REGISTRY_SETTING
 description: "FakeAlert.BG trojan check"
 value_type: POLICY_TEXT
 reg_key: "HKU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
 reg_item: "brastk"
 value_data: "C:\WINDOWS\System32\brastk.exe"
 reg_option: CAN_BE_NULL
 check_type: CHECK_NOT_EQUAL
 info: "A registry entry for FakeAlert.BG trojan/downloader was found."
 info: "The contents of this audit can be edited as desired."
</custom_item>
```

Les types de champ **value_type** principaux suivants sont disponibles :

- POLICY_SET
  value_data: "Enabled" ou "Disabled"
- POLICY_DWORD
  value_data: DWORD ou RANGE [même dword que pour registry ou range]
- POLICY_TEXT
  value_data: "TEXT" [même texte que pour registry]
- POLICY_MULTI_TEXT
  value_data: "TEXT1" && "TEXT2" && ... && "TEXTN" [mêmes textes que pour registry]
- POLICY_BINARY
  value_data: "0102ac0b...34fb" [même binaire que pour registry]
- FILE_ACL, REG_ACL, SERVICE_ACL, LAUNCH_ACL, ACCESS_ACL
  value_data: "acl_name" [nom de l'acl à utiliser]

Les types de champ **value_type** facultatifs suivants sont disponibles et utilisés dans les éléments prédéfinis :

- DRIVER_SET
  value_data: "Silent Succeed", "Warn but allow installation", "Do not allow installation"
- LDAP_SET
  value_data: "None" ou "Require Signing"
- LOCKEDID_SET
  value_data: "user display name, domain and user names", "user display name only", "do not display user information"
- SMARTCARD_SET
  value_data: "No action", "Lock workstation", "Force logoff", "Disconnect if a remote terminal services session"
- LOCALACCOUNT_SET
  value_data: "Classic - local users authenticate as themselves", "Guest only - local users authenticate as guest"
- NTLMSSP_SET
  value_data: "No minimum", "Require message integrity", "Require message confidentiality", "Require ntlmv2 session security", "Require 128-bit encryption"

- CRYPTO_SET
  value_data: "User input is not required when new keys are stored and used", "User is prompted when the key is first used" ou "User must enter a password each time they use a key"

- OBJECT_SET
  value_data: "Administrators group", "Object creator"

- DASD_SET
  value_data: "Administrators", "administrators and power users", "Administrators and interactive users"

- LANMAN_SET
  value_data: "Send LM & NTLM responses", "send lm & ntlm - use ntlmv2 session security if negotiated", "send ntlm response only", "send ntlmv2 response only", "send ntlmv2 response only\refuse lm" ou "send ntlmv2 response only\refuse lm & ntlm"

- LDAPCLIENT_SET
  value_data: "None", "Negotiate Signing" ou "Require Signing"

- EVENT_METHOD
  value_data: "by days", "manually" ou "as needed"

- POLICY_DAY
  value_data: DWORD ou RANGE (durée en jours)

- POLICY_KBYTE
  value_data: DWORD ou RANGE

Pour le champ **custom_item**, utilisez le **value_type** principal. Des types facultatifs ont été créés pour les éléments prédéfinis.

Si le **value_type** est une ACL, l'élément de registre doit être une description de sécurité au format binaire.

Exemples :

```
<custom_item>
 type: REGISTRY_SETTING
 description: "Network security: Do not store LAN Manager hash value on next password
       change"
 value_type: POLICY_SET
 value_data: "Enabled"
 reg_key: "HKLM\SYSTEM\CurrentControlSet\Control\Lsa"
 reg_item: "NoLMHash"
</custom_item>
```

```
<custom_item>
 type: REGISTRY_SETTING
 description: "Network access: Shares that can be accessed anonymously"
 value_type: POLICY_MULTI_TEXT
 value_data: "SHARE" && "EXAMPLE$"
 reg_key: "HKLM\SYSTEM\CurrentControlSet\Services\LanManServer\Parameters"
 reg_item: "NullSessionShares"
</custom_item>
```

```
<custom_item>
 type: REGISTRY_SETTING
 description: "DCOM: Network Provisioning Service - Launch permissions"
 value_type: LAUNCH_ACL
 value_data: "2"
 reg_key: "HKLM\SOFTWARE\Classes\AppID\{39ce474e-59c1-4b84-9be2-2600c335b5c6}"
 reg_item: "LaunchPermission"
</custom_item>
```

```
<custom_item>
 type: REGISTRY_SETTING
 description: "DCOM: Automatic Updates - Access permissions"
 value_type: ACCESS_ACL
 value_data: "3"
 reg_key: "HKLM\SOFTWARE\Classes\AppID\{653C5148-4DCE-4905-9CFD-1B23662D3D9E}"
 reg_item: "AccessPermission"
</custom_item>
```

REGISTRY_PERMISSIONS

Le bon fonctionnement de ce contrôle nécessite que le système Windows distant ait accès au registre distant.

Utilisation

```
<custom_item>
 type: REGISTRY_PERMISSIONS
 description: ["description"]
 value_type: [value_type]
 value_data: [value]
 (optional) check_type: [value]
 reg_key: ["regkeyname"]
 (optional) acl_option: [acl_option]
</custom_item>
```

Cet élément de stratégie vérifie si l'ACL de clé de registre est correcte. Le contrôle est effectué en invoquant la fonction **`RegGetKeySecurity`** sur l'identificateur de clé de registre.

Le type autorisé est :

```
value_type: REG_ACL
value_data: "ACLname"
reg_key: "RegistryKeyName"
```

Les chemins prédéfinis suivants peuvent être utilisés pour le champ **`reg_key`** :

```
HKLM (HKEY_LOCAL_MACHINE)
HKU (HKEY_USERS)
HKCR (HKEY_CLASS_ROOT)
```

Lorsque cet audit est utilisé, il faut noter les points suivants :

- Le champ **reg_key** doit inclure le chemin complet de la clé de registre du fichier.
- Le champ **value_data** est le nom d'une ACL définie dans le fichier de la stratégie.
- Le champ **acl_option** peut être paramétré sur CAN_BE_NULL (peut être nul) ou CAN_NOT_BE_NULL (ne peut pas être nul) pour forcer une réussite/erreur si la clé n'existe pas

Exemple :

```
 <registry_acl: "ACL2">

  <user: "Administrators">
   acl_inheritance: "not inherited"
   acl_apply: "This key and subkeys"
   acl_allow: "Full Control"
  </user>

  <user: "SYSTEM">
   acl_inheritance: "not inherited"
   acl_apply: "This key and subkeys"
   acl allow: "Full Control"
  </user>

 </acl>

 <custom item>
  type: REGISTRY_PERMISSIONS
  description: "Permissions for HKLM\SOFTWARE\Microsoft"
  value_type: REG_ACL
  value_data: "ACL2"
  reg_key: "HKLM\SOFTWARE\Microsoft"
 </custom_item>
```

Lorsque le contrôle ci-dessus est exécuté, le module de conformité vérifie si les permissions définies pour "**HKLM\SOFTWARE\Microsoft** » correspondent à celles décrites dans registry_acl ACL2.

REGISTRY_AUDIT

Le bon fonctionnement de ce contrôle nécessite que le système Windows distant ait accès au registre distant.

Utilisation

```
<custom_item>
 type: REGISTRY_AUDIT
 description: ["description"]
 value_type: [value_type]
 value_data: [value]
 reg_key: ["regkeyname"]
 (optional) acl_option: [acl_option]
</custom_item>
```

Cet élément de stratégie vérifie si l'ACL de clé de registre est correcte. Le contrôle est effectué en invoquant la fonction **`RegGetKeySecurity`** sur l'identificateur de clé de registre.

Le type autorisé est :

```
value_type: REG_ACL
value_data: "ACLname"
reg_key: "RegistryKeyName"
```

Le chemin prédéfini suivant peut être utilisé pour le champ **`reg_key`** field:

```
HKLM (HKEY_LOCAL_MACHINE)
HKU (HKEY_USERS)
HKCR (HKEY_CLASS_ROOT)
```

Lorsque cet audit est utilisé, il faut noter les points suivants :

- Le champ **`reg_key`** doit inclure le chemin complet de la clé de registre du fichier.
- Le champ **`value_data`** est le nom de l'ACL définie dans le fichier de la stratégie.
- Le champ The **`acl_option`** peut être paramétré sur CAN_BE_NULL (peut être nul) ou CAN_NOT_BE_NULL (ne peut pas être nul) pour forcer une réussite/erreur si la clé n'existe pas.
- Les champs **`acl_allow`** et **`acl_deny`** correspondent aux événements d'audit « Successful » (Réussite) et « Failed » (Échec).

Cet exemple de fichier **`.audit`** vérifie la clé de registre de « HKLM\SOFTWARE\Microsoft » en fonction d'une liste de contrôle d'accès nommée « ACL2 » qui n'est pas montrée :

```
 <custom_item>
  type: REGISTRY_AUDIT
  description: "Audit for HKLM\SOFTWARE\Microsoft"
  value_type: REG_ACL
  value_data: "ACL2"
  reg_key: "HKLM\SOFTWARE\Microsoft"
 </custom_item>
```

REGISTRY_TYPE

Le bon fonctionnement de ce contrôle nécessite que le système Windows distant ait accès au registre distant.

Utilisation

```
<custom_item>
 type: REGISTRY_TYPE
 description: ["description"]
 value_type: [VALUE_TYPE]
 value_data: [value]
 reg_key: ["key name"]
 reg_item: ["key item"]
 (optional) reg_option: [KEY_OPTIONS]
</item>
```

Cet élément de stratégie est utilisé pour contrôler la valeur d'un type de clé de registre. Le contrôle est effectué en invoquant la fonction **`RegQueryValue`**.

Le champ **`reg_key`** fest le nom de la clé de registre (par exemple « HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon »). La première partie de la clé (HKLM, HKU, HKCU, ...) est utilisée pour la connexion à la ruche du registre correcte. Dans la plupart des cas, le champ **`reg_key`** exige une entrée de registre statique sans caractères génériques ; cependant, une exception est autorisée pour la recherche des valeurs dans HKU (HKEY_USERS). Si un chemin est désigné sous HKU, la recherche itère sur toutes les valeurs utilisateur de HKU pour rechercher la valeur placée sous le chemin désigné. Par exemple, si `reg_key: "HKU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"` est spécifié avec **`reg_item`** "brastk", le programme recherchera la valeur de la clé de registre "brastk" pour tous les utilisateurs sous HKU sous le chemin relatif : “HKU\<user_id>\SOFTWARE\Microsoft\Windows\CurrentVersion\Run”. Par exemple :

```
value_type: POLICY_TEXT
reg_key: "HKU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
reg_item: "brastk"
value_data: "C:\WINDOWS\System32\brastk.exe"
```

Ce contrôle effectue la recherche sous :

HKU\S-1-5-18\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKU\S-1-5-19\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Le champ facultatif **`reg_option`** peut être défini avec la valeur CAN_BE_NULL pour forcer la réussite du contrôle si la clé n'existe pas, ou avec la valeur opposée CAN_NOT_BE_NULL.

Seule la valeur de champ **`value_type`** POLICY_TEXT est disponible pour ce contrôle. .

Dans cet exemple, le fichier **`.audit`** vérifie le type de registre de « HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon » :

```
<custom_item>
 type: REGISTRY_TYPE
 description: "Check type - reg_sz"
 value_type: POLICY_TEXT
 value_data: "reg_sz"
 reg key: "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon"
 reg_item: "ScreenSaverGracePeriod"
</item>
```

Veuillez noter que la vérification de HKCU peut être impossible sur la plupart des installations Windows. En effet, cette vérification exige les clés « Currrent user » (Utilisateur actuel), qui n'existent normalement pas lorsque Nessus effectue l'authentification via SMB. La vérification HKU (tous les utilisateurs) permet de résoudre ce problème. Lorsque le plugin détecte qu'une clé HKU fait l'objet d'une vérification, il effectue automatiquement une boucle sur tous les SID disponibles, à l'exception de la clé `.DEFAULT`. Cette approche présente l'inconvénient de vérifier également les utilisateurs système (par exemple, SYSTEM, NT Authority, etc.). Utilisez `reg_ignore_hku_users` si vous ne voulez pas vérifier ces utilisateurs. Par exemple :

```
reg_ignore_hku_users : "S-1-5-18,S-1-5-19,S-1-5-20"
```

Cette commande est uniquement possible avec la vérification `REGISTRY_SETTING`.

SERVICE_PERMISSIONS

Utilisation

```
<custom_item>
 type: SERVICE_PERMISSIONS
 description: ["description"]
 value_type: [value_type]
 value_data: [value]
 (optional) check_type: [value]
 service: ["servicename"]
 (optional) acl_option: [acl_option]
</custom_item>
```

Cet élément de stratégie vérifie si l'ACL de service est correcte. Le contrôle est effectué en invoquant la fonction **`QueryServiceObjectSecurity`** sur l'identificateur de service.

Le type autorisé est :

```
value_type: SERVICE_ACL
value_data: "ACLname"
service: "ServiceName"
```

Lorsque cet audit est utilisé, il faut noter les points suivants :

- Le champ **`value_data`** est le nom d'une ACL définie dans le fichier de la stratégie.
- Le champ **`acl_option`** peut être paramétré sur CAN_BE_NULL (peut être nul) ou CAN_NOT_BE_NULL (ne peut pas être nul) pour forcer une réussite/erreur si la clé n'existe pas.

Exemple :

```
 <service_acl: "ACL3">

  <user: "Administrators">
   acl_inheritance: "not inherited"
   acl_apply: "This object only"
   acl_allow: "query template" | "change template" | "query status" | "enumerate
        dependents" | "start" | "stop" | "pause and continue" | "interrogate" | "user-
        defined control" | "delete" | "read permissions" | "change permissions" | "take
        ownership"
  </user>

  <user: "SYSTEM">
   acl_inheritance: "not inherited"
   acl_apply: "This object only"
   acl_allow: "query template" | "change template" | "query status" | "enumerate
        dependents" | "start" | "stop" | "pause and continue" | "interrogate" | "user-
        defined control" | "delete" | "read permissions" | "change permissions" | "take
        ownership"
  </user>

  <user: "Interactive">
   acl_inheritance: "not inherited"
   acl_apply: "This object only"
```

```
   acl_allow: "query template" | "query status" | "enumerate dependents" |
        "interrogate" | "user-defined control" | "read permissions"
  </user>

  <user: "Everyone">
   acl_inheritance: "not inherited"
   acl_apply: "This object only"
   acl_allow: "query template" | "change template" | "query status" | "enumerate
        dependents" | "start" | "stop" | "pause and continue" | "interrogate" | "user-
        defined control" | "delete" | "read permissions" | "change permissions" | "take
        ownership"
  </user>

 </acl>

 <custom_item>
  type: SERVICE_PERMISSIONS
  description: "Permissions for Alerter Service"
  value_type: SERVICE_ACL
  value_data: "ACL3"
  service: "Alerter"
 </custom_item>
```

Lorsque le contrôle ci-dessus est exécuté, le module de conformité vérifie si les permissions définies pour le service Alerter correspondent à celles décrites dans service_acl ACL3.

SERVICE_AUDIT

Utilisation

```
<custom_item>
 type: SERVICE_AUDIT
 description: ["description"]
 value_type: [value_type]
 value_data: [value]
 (optional) check_type: [value]
 service: ["servicename"]
 (optional) acl_option: [acl_option]
</custom_item>
```

Cet élément de stratégie vérifie si l'ACL de service est correcte. Le contrôle est effectué en invoquant la fonction **`QueryServiceObjectSecurity`** sur l'identificateur de service.

Le type autorisé est :

```
value_type: SERVICE_ACL
value_data: "ACLname"
service: "ServiceName"
```

Lorsque ce type d'audit est utilisé, il faut noter les points suivants :

- Le champ **`value_data`** est le nom de l'ACL définie dans le fichier de la stratégie.
- Le champ **`acl_option`** peut être paramétré sur CAN_BE_NULL (peut être nul) ou CAN_NOT_BE_NULL (ne peut pas être nul) pour forcer une réussite/erreur si la clé n'existe pas.

- Les champs **acl_allow** et **acl_deny** correspondent aux événements d'audit « Successful » (Réussite) et « Failed » (Échec).

Voici un exemple de fichier **.audit** permettant de vérifier le service « Alerter » :

```
 <custom_item>
  type: SERVICE_AUDIT
  description: "Audit for Alerter Service"
  value_type: SERVICE_ACL
  value_data: "ACL3"
  service: "Alerter"
 </custom_item>
```

WMI_POLICY

Utilisation

```
<custom_item>
 type: WMI_POLICY
 description: "Test for WMI Value"
 value_type: [value_type]
 value_data: [value]
 (optional) check_type: [value]
 wmi_namespace: ["namespace"]
 wmi_request: ["request select statement"]
 wmi_attribute: ["attribute"]
 wmi_key: ["key"]
</custom_item>
```

Ce contrôle interroge la base de données WMI de Windows pour les valeurs spécifiées dans espace de noms/classe/attribut.

Les valeurs clés peuvent être extraites ou les noms d'attribut peuvent être énumérés, selon la syntaxe utilisée.

Les types autorisés sont :

```
wmi_namespace: "namespace"
wmi_request: "WMI Query"
wmi_attribute: "Name"
wmi_key: "Name"
wmi_option: option
wmi_exclude_result: "result"
only_show_query_output: YES
check_type: CHECK_NOT_REGEX
```

Si vous effectuez une sélection à partir d'une configuration de service avec valeurs dupliquées sur le système (par exemple « MSFTPSVC/83207416 » et « MSFTPSVC/2 »), la demande extraira l'attribut choisi pour les deux. Si l'une des deux ne correspond pas à la valeur de la stratégie, **wmi_key** sera ajouté au rapport pour indiquer celle qui a échoué. Le champ **wmi_enum** permet d'énumérer les noms de configuration dans un espace de noms pour servir de comparaison ou pour vérifier les valeurs de stratégie.

Par défaut, si une interrogation WMI ne renvoie aucune sortie, le contrôle signale une erreur. Ce comportement peut être modifié et le contrôle peut être forcé de signaler un résultat concluant (PASS) si **wmi_option** est défini avec la valeur CAN_BE_NULL. Si **only_show_query_output** a la valeur YES (OUI), la sortie de l'interrogation WMI est désormais

incluse dans le rapport Nessus. Avec la balise **`check_type`**, vous pouvez obtenir un résultat concluant (PASS) tant que la sortie ne contient pas une chaîne spécifique. Voir les exemples ci-dessous.

Autres considérations :

- Les attributs WMI doivent être spécifiés explicitement. Par exemple, `select * from foo` ne donnera aucun résultat.
- Les attributs pour lesquels aucune valeur n'est définie ne seront pas signalés.
- La casse des attributs doit correspondre exactement aux spécifications de la documentation Microsoft. Par exemple, l'attribut `HandleCount` ne peut pas être représenté par `Handlecount` ou `handlecount`.
- Les valeurs de type tableau ne sont pas incluses dans le résultat.

Exemple 1 :

```
<custom_item>
 type: WMI_POLICY
 description: "IIS test"
 value_type: POLICY_DWORD
 value_data: 0
 wmi_namespace: "root/MicrosoftIISv2"
 wmi_request: "SELECT Name, UserIsolationMode FROM IIsFtpServerSetting"
 wmi_attribute: "UserIsolationMode"
 wmi_key: "Name"
</custom_item>
```

S'il existe deux configurations de service FTP sur le système (« MSFTPSVC/83207416 » et « MSFTPSVC/2 »), la demande extrait l'attribut « UserIsolationMode » à partir des deux. Si l'une des deux ne correspond pas la valeur de la stratégie (0), **`wmi_key`** (dans ce cas) sera ajouté au rapport en indiquant celle qui a échoué.

Exemple 2 :

```
<custom item>
 type: WMI_POLICY
 description: "IIS test2"
 value_type: POLICY_MULTI_TEXT
 value_data: "MSFTPSVC/83207416" && "MSFTPSVC/2"
 wmi namespace: "root/MicrosoftIISv2"
 wmi_request: "SELECT Name FROM IIsFtpServerSetting"
 wmi_attribute: "Name"
 wmi_key: "Name"
 wmi_option: WMI_ENUM
</custom_item>
```

Cet exemple vérifie qu'il existe deux noms de configuration valides comme spécifié dans **`value_data`**. Si vous souhaitez en savoir davantage sur l'espace de noms WMI et les attributs associés, le WMI CIM Studio de Microsoft est un outil utile disponible à l'adresse suivante :

http://www.microsoft.com/downloads/details.aspx?FamilyID=6430f853-1120-48db-8cc5-f2abdc3ed314&displaylang=en (en anglais)

Exemple 3 :

```
<custom_item>
 type: WMI_POLICY
 description: "List All Windows Processes - except svchost.exe and iPodService.exe"
 value_type: POLICY_TEXT
 value_data: ""
 wmi_namespace: "root/cimv2"
 wmi_exclude_result: "svchost.exe,iPodService.exe"
 wmi_request: "select Caption,HandleCount,ThreadCount from Win32_Process"
 only_show_query_output: YES
</custom_item>
```

Cet exemple répertorie tous les processus Windows, mais il supprime les instances de **`svchost.exe`** et de **`iPodService.exe`**.

Éléments

Les « Éléments » sont des types de contrôle qui sont prédéfinis dans Windows Compliance Checks Engine (moteur d'audit de conformité Windows). Ils sont utilisés pour les éléments couramment vérifiés et réduisent au minimum la syntaxe requise pour la création des contrôles d'audit. Un élément a la structure suivante :

```
<item>
 name: ["predefined_entry"]
 value: [value]
</item>
```

Le champ **`name`** doit avoir un nom qui est déjà défini (les noms prédéfinis sont répertoriés dans le tableau « Stratégies prédéfinies » ci-dessous).

Tous les éléments prédéfinis correspondent à la liste disponible dans l'Éditeur de stratégie de domaine sur Windows 2003 SP1.

L'exemple suivant vérifie si la longueur minimale du mot de passe est comprise entre 8 et 14 caractères :

```
<item>
 name: "Minimum password length"
 value: [8..14]
</item>
```

L'élément personnalisé correspondant est :

```
<custom_item>
 type: PASSWORD_POLICY
 description: "Minimum password length"
 value_type: POLICY_DWORD
 value_data: [8..14]
 password_policy: MINIMUM_PASSWORD_LENGTH
</custom_item>
```

Stratégies prédéfinies

Stratégie	Utilisation
Password Policy (Stratégie de mot de passe)	name: "Enforce password history" value: POLICY_DWORD name: "Maximum password age" value: TIME_DAY name: "Minimum password age" value: TIME_DAY name: "Minimum password length" value: POLICY_DWORD name: "Password must meet complexity requirements" value: POLICY_SET
Account Lockout Policy (Stratégie de verrouillage de compte)	name: "Account lockout duration" value: TIME_MINUTE ou name: "Account lockout duration" value: TIME_SECOND name: "Account lockout threshold" value: POLICY_DWORD name: "Reset lockout account counter after" value: TIME_MINUTE name: "Enforce user logon restrictions" value: POLICY_SET
Kerberos Policy (Stratégie Kerberos)	name: "Maximum lifetime for service ticket" value: TIME_MINUTE name: "Maximum lifetime for user ticket" value: TIME_HOUR name: "Maximum lifetime for user renewal ticket" value: TIME_DAY name: "Maximum tolerance for computer clock synchronization" value: TIME_MINUTE
Audit Policy (Stratégie d'audit)	name: "Audit account logon events" value: AUDIT_SET name: "Audit account management" value: AUDIT_SET name: "Audit directory service access" value: AUDIT_SET name: "Audit logon events" value: AUDIT_SET

	name: "Audit object access" value: AUDIT_SET name: "Audit policy change" value: AUDIT_SET name: "Audit privilege use" value: AUDIT_SET name: "Audit process tracking" value: AUDIT_SET name: "Audit system events" value: AUDIT_SET
Accounts (Comptes)	name: "Accounts: Administrator account status" value: POLICY_SET name: "Accounts: Guest account status" value: POLICY_SET name: "Accounts: Limit local account use of blank password to console logon only" value: POLICY_SET name: "Accounts: Rename administrator account" value: POLICY_TEXT name: "Accounts: Rename guest account" value: POLICY_TEXT
Audit (Audit)	name: "Audit: Audit the access of global system objects" value: POLICY_SET name: "Audit: Audit the use of Backup and Restore privilege" value: POLICY_SET name: "Audit: Shut down system immediately if unable to log security audits" value: POLICY_SET
DCOM (DCOM)	name: "DCOM: Machine Launch Restrictions in Security Descriptor Definition Language (SDDL) syntax" value: POLICY_TEXT name: "DCOM: Machine Access Restrictions in Security Descriptor Definition Language (SDDL) syntax" value: POLICY_TEXT
Devices (Périphériques)	name: "Devices: Allow undock without having to log on" value: POLICY_SET name: "Devices: Allowed to format and eject removable media" value: DASD_SET name: "Devices: Prevent users from installing printer drivers" value: POLICY_SET

	name: "Devices: Restrict CD-ROM access to locally logged-on user only" value: POLICY_SET name: "Devices: Restrict floppy access to locally logged-on user only" value: POLICY_SET name: "Devices: Unsigned driver installation behavior" value: DRIVER_SET
Domain controller (Contrôleur de domaine)	name: "Domain controller: Allow server operators to schedule tasks" value: POLICY_SET name: "Domain controller: LDAP server signing requirements" value: LDAP_SET name: "Domain controller: Refuse machine account password changes" value: POLICY_SET
Domain member (Membre de domaine)	name: "Domain member: Digitally encrypt or sign secure channel data (always)" value: POLICY_SET name: "Domain member: Digitally encrypt secure channel data (when possible)" value: POLICY_SET name: "Domain member: Digitally sign secure channel data (when possible)" value: POLICY_SET name: "Domain member: Disable machine account password changes" value: POLICY_SET name: "Domain member: Maximum machine account password age" value: POLICY_DAY name: "Domain member: Require strong (Windows 2000 or later) session key" value: POLICY_SET
Interactive logon (Connexion interactive)	name: "Interactive logon: Display user information when the session is locked" value: LOCKEDID_SET name: "Interactive logon: Do not display last user name" value: POLICY SET name: "Interactive logon: Do not require CTRL+ALT+DEL" value: POLICY_SET name: "Interactive logon: Message text for users attempting to log on"

	value: POLICY_TEXT name: "Interactive logon: Message title for users attempting to log on" value: POLICY_TEXT name: "Interactive logon: Number of previous logons to cache (in case domain controller is not available)" value: POLICY_DWORD name: "Interactive logon: Prompt user to change password before expiration" value: POLICY_DWORD name: "Interactive logon: Require Domain Controller authentication to unlock workstation" value: POLICY_SET name: "Interactive logon: Require smart card" value: POLICY_SET name: "Interactive logon: Smart card removal behavior" value: SMARTCARD_SET
Microsoft network client (Client réseau Microsoft)	name: "Microsoft network client: Digitally sign communications (always)" value: POLICY_SET name: "Microsoft network client: Digitally sign communications (if server agrees)" value: POLICY_SET name: "Microsoft network client: Send unencrypted password to third-party SMB servers" value: POLICY_SET
Microsoft network server (Serveur réseau Microsoft)	name: "Microsoft network server: Amount of idle time required before suspending session" value: POLICY_DWORD name: "Microsoft network server: Digitally sign communications (always)" value: POLICY_SET name: "Microsoft network server: Digitally sign communications (if client agrees)" value: POLICY_SET name: "Microsoft network server: Disconnect clients when logon hours expire" value: POLICY_SET
Network access (Accès réseau)	name: "Network access: Allow anonymous SID/Name translation" value: POLICY_SET name: "Network access: Do not allow anonymous enumeration of SAM accounts"

	value: POLICY_SET name: "Network access: Do not allow anonymous enumeration of SAM accounts and shares" value: POLICY_SET name: "Network access: Do not allow storage of credentials or .NET Passports for network authentication" value: POLICY_SET name: "Network access: Let Everyone permissions apply to anonymous users" value: POLICY_SET name: "Network access: Named Pipes that can be accessed anonymously" value: POLICY_MULTI_TEXT name: "Network access: Remotely accessible registry paths and sub-paths" value: POLICY_MULTI_TEXT name: "Network access: Remotely accessible registry paths" value: POLICY_MULTI_TEXT name: "Network access: Restrict anonymous access to Named Pipes and Shares" value: POLICY_SET name: "Network access: Shares that can be accessed anonymously" value: POLICY_MULTI_TEXT name: "Network access: Sharing and security model for local accounts" value: LOCALACCOUNT_SET
Network security (Sécurité réseau)	name: "Network security: Do not store LAN Manager hash value on next password change" value: POLICY_SET name: "Network security: Force logoff when logon hours expire" value: POLICY_SET name: "Network security: LAN Manager authentication level" value: LANMAN_SET name: "Network security: LDAP client signing requirements" value: LDAPCLIENT_SET name: "Network security: Minimum session security for NTLM SSP based (including secure RPC) clients" value: NTLMSSP_SET name: "Network security: Minimum session security for NTLM SSP based (including secure RPC) servers" value: NTLMSSP_SET

Recovery console (Console de récupération)	name: "Recovery console: Allow automatic administrative logon" value: POLICY_SET name: "Recovery console: Allow floppy copy and access to all drives and all folders" value: POLICY_SET
Shutdown (Arrêt)	name: "Shutdown: Allow system to be shut down without having to log on" value: POLICY_SET name: "Shutdown: Clear virtual memory pagefile" value: POLICY_SET
System cryptography (Chiffrement système)	name: "System cryptography: Force strong key protection for user keys stored on the computer" value: CRYPTO_SET name: "System cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing" value: POLICY_SET
System objects (Objets système)	name: "System objects: Default owner for objects created by members of the Administrators group" value: OBJECT_SET name: "System objects: Require case insensitivity for non-Windows subsystems" value: POLICY_SET name: "System objects: Strengthen default permissions of internal system objects (e.g. Symbolic Links)" value: POLICY_SET
System settings (Paramètres système)	name: "System settings: Optional subsystems" value: POLICY_MULTI_TEXT name: "System settings: Use Certificate Rules on Windows Executables for Software Restriction Policies" value: POLICY_SET
Event Log (Journal d'événements)	name: "Maximum application log size" value: POLICY_KBYTE name: "Maximum security log size" value: POLICY_KBYTE name: "Maximum system log size" value: POLICY_KBYTE name: "Prevent local guests group from accessing application log" value: POLICY_SET name: "Prevent local guests group from accessing security log" value: POLICY_SET name: "Prevent local guests group from accessing system log"

```
value: POLICY_SET

name: "Retain application log"
value: POLICY_DAY

name: "Retain security log"
value: POLICY_DAY

name: "Retain system log"
value: POLICY_DAY

name: "Retention method for application log"
value: EVENT_METHOD

name: "Retention method for security log"
value: EVENT_METHOD

name: "Retention method for system log"
value: EVENT_METHOD
```

Rapports forcés

L'utilisation du mot clé « **report** » permet de forcer les stratégies d'audit à produire un résultat spécifique. Les types de rapport PASSED (SUCCÈS), FAILED (ÉCHEC) et WARNING (AVERTISSEMENT) peuvent être utilisés. Un exemple de stratégie est fourni ci-dessous :

```
<report type: "WARNING">
 description: "Audit 103-a requires a physical inspection of the pod bay doors Hal"
</report>
```

Le texte dans le champ « **description** » est toujours affiché dans le rapport.

Ce type de rapport est utile si vous souhaitez informer un auditeur qu'il est impossible de compléter un contrôle en cours d'exécution par Nessus. Par exemple, il peut être nécessaire de déterminer si un système spécifique a été physiquement sécurisé et nous souhaitons informer l'auditeur d'effectuer le contrôle ou l'inspection manuellement. Ce type de rapport est aussi utile si le type de vérification spécifié qui doit être effectué par Nessus n'a pas été déterminé par un contrôle OVAL.

Conditions

Il est possible de définir la logique **if/then/else** dans la stratégie Windows pour démarrer un contrôle seulement si les conditions préalables sont valides ou pour regrouper plusieurs tests en un seul.

La syntaxe pour exécuter les conditions est la suivante :

```
<if>
 <condition type: "or">
  <Insert your audit here>
 </condition>
 <then>
  <Insert your audit here>
 </then>
 <else>
  <Insert your audit here>
 </else>
</if>
```

Les conditions peuvent être du type « **and** » ou « **or** ».

L'audit pour la condition, les déclarations « **then** » et « **else** » peuvent être une liste d'éléments (ou élément personnalisés) ou une déclaration « **if** ». Les déclarations « **else** » et « **then** » peuvent aussi utiliser le type « **report** » pour signaler une réussite ou un échec selon la valeur de retour de la condition :

```
<report type:"PASSED|FAILED">
 description: "the test passed (or failed)"
 (optional) severity: INFO|MEDIUM|HIGH
</report>
```

Une valeur « **if** » renvoie SUCCESS (Réussite) ou FAILURE (Échec) et cette valeur est utilisée lorsque la déclaration « **if** » est à l'intérieur d'une autre structure « **if** ». Par exemple, si la structure **<then>** est exécutée, la valeur renvoyée sera l'une des suivantes :

- l'audit contient les éléments : renvoie SUCCESS (Réussite) si tous les éléments ont été contrôlés avec succès, sinon renvoie FAILURE (Échec)
- l'audit contient uniquement **<report>** : renvoie le type de rapport
- l'audit contient les éléments et **<report>** : renvoie le type de rapport

Si la déclaration **<report>** est utilisée et le type est « FAILED » (Échec), la raison de l'échec est indiquée dans le rapport ainsi que le degré de sévérité s'il est défini.

Voici un exemple d'audit de la stratégie de mot de passe. Puisque le type « **and** » est utilisé, les deux éléments personnalisés doivent passer l'audit avec succès pour que cette stratégie passe avec succès. Cet exemple met à l'essai une combinaison très inhabituelle de stratégies historiques de mots de passe valides pour illustrer la mise en œuvre d'une logique de test sophistiquée.

```
<if>
 <condition type:"and">
  <custom_item>
   type: PASSWORD_POLICY
   description: "2.2.2.5 Password History: 24 passwords remembered"
   value_type: POLICY_DWORD
   value_data: [22..MAX] || 20
   password_policy: ENFORCE_PASSWORD_HISTORY
  </custom_item>
  <custom_item>
   type: PASSWORD_POLICY
   description: "2.2.2.5 Password History: 24 passwords remembered"
   value_type: POLICY_DWORD
   value_data: 18 || [4..24]
   password_policy: ENFORCE_PASSWORD_HISTORY
  </custom_item>
 </condition>

 <then>
  <report type:"PASSED">
   description: "Password policy passed"
  </report>
 </then>

 <else>
  <report type:"FAILED">
   description: "Password policy failed"
```

```
  </report>
 </else>
</if>
```

Dans l'exemple ci-dessus, seul le nouveau type « **report** » a été montré mais la structure **if/then/else** » prend en charge l'exécution d'audits supplémentaires dans le cadre des clauses « else ». Au sein d'une condition, des clauses if/then/else imbriquées peuvent aussi être utilisées. Un exemple plus complexe est montré ci-dessous :

```
<if>
 <condition type:"and">
  <custom_item>
   type: CHECK_ACCOUNT
   description: "Accounts: Rename Administrator account"
   value_type: POLICY_TEXT
   value_data: "Administrator"
   account_type: ADMINISTRATOR_ACCOUNT
   check_type: CHECK_NOT_EQUAL
  </custom_item>
 </condition>

 <then>
  <report type:"PASSED">
   description: "Administrator account policy passed"
  </report>
 </then>

 <else>
  <if>
   <condition type:"or">
    <item>
    name: "Minimum password age"
    value: [1..30]
    </item>
    <custom_item>
     type: PASSWORD_POLICY
     description: "Password Policy setting"
     value_type: POLICY_SET
     value_data: "Enabled"
     password_policy: COMPLEXITY_REQUIREMENTS
    </custom_item>
 </condition>

   <then>
    <report type:"PASSED">
     description: "Administrator account policy passed"
    </report>
   </then>

   <else>
    <report type:"FAILED">
     description: "Administrator account policy failed"
    </report>
   </else>
  </if>
```

```
  </else>
 </if>
```

Dans cet exemple, si le compte d'administrateur n'a pas été renommé, vérifiez que l'antériorité minimale du mot de passe est 30 jours ou moins. Cette stratégie de vérification passe avec succès si le compte d'administrateur a été renommé, quelle que soit la stratégie de mot de passe, et teste la stratégie d'antériorité du mot de passe seulement si le compte d'administrateur n'a pas été renommé.

Référence des fichiers de conformité d'audit pour le contenu Windows

Les contrôles du contenu de Windows **`.audit`** sont différents des contrôles de la configuration de Windows Configuration **`.audit`** car ils sont conçus pour rechercher dans un système de fichiers Windows des types de fichier spécifiques contenant des données sensibles au lieu d'énumérer les paramètres de configuration du système. Ils incluent une série d'options pour aider l'auditeur à préciser les paramètres de recherche, à localiser plus efficacement les données non conformes et à les afficher.

Emploi des guillemets :

Les guillemets simples et doubles sont interchangeables autour des champs d'audit, sauf dans les deux cas suivants :

1. Dans les contrôles de conformité Windows où des champs spéciaux tels que CRLF, doivent être interprétés littéralement, utilisez des guillemets simples. Tout champ intégré qui doit être interprété comme une chaîne doit être inclus dans une séquence d'échappement.

Par exemple :
```
expect: 'First line\r\nSecond line\r\nJohn\'s Line'
```

2. Des guillemets doubles sont requis lorsque vous utilisez les fichiers Windows « include_paths » et « exclude_paths ».

Si des chaînes sont utilisées dans tout type de champ (description, value_data, regex, etc.) contenant des guillemets simples ou doubles, il y a deux façons de les traiter :

a. Utilisez le type de guillemets opposés pour les guillemets extérieurs.

Par exemple :
```
expect: "This is John's Line"
expect: 'We are looking for a double-quote-".*'
```

b. Échappez tout guillemet intégré avec une barre oblique inverse (guillemets doubles uniquement).

Par exemple :
```
expect: "\"Text to be searched\""
```

Type de contrôle

Tous les contrôles de conformité du contenu de Windows doivent être mis entre crochets avec l'encapsulation **`check_type`** et la désignation « **`WindowsFiles`** ». Ce point est très semblable à tous les autres fichiers **`.audit`**. Le format de base d'un fichier de contrôle de contenu est le suivant :

```
<check_type: "WindowsFiles">
<item>
</item>
<item>
</item>
```

```
<item>
</item>
</check_type>
```

Les contrôles réels pour chaque élément ne sont pas affichés. Les sections ci-dessous montrent comment divers mots clés et paramètres peuvent être utilisés pour remplir un audit spécifique d'élément de contenu.

Format d'élément

Utilisation
<item> type: FILE_CONTENT_CHECK description: ["value data"] file_extension: ["value data"] *(optional)* regex: ["value data"] *(optional)* expect: ["value data"] *(optional)* file_name: ["value data"] *(optional)* max_size: ["value data"] *(optional)* only_show: ["value data"] *(optional)* regex_replace: ["value data"] </item>

Chacun de ces éléments est utilisé pour vérifier une large gamme de formats de fichiers, avec une grande variété de types de données. Le tableau ci-dessous fournit une liste des types de données pris en charge. La section suivante montre de nombreux exemples pour l'utilisation conjointe de ces mots clés afin de vérifier divers types de contenus de fichier.

Mot clé	Description
type	Doit toujours être paramétré sur FILE_CONTENT_CHECK
description	Information qui sera utilisée comme titre pour les vulnérabilités uniques de conformité dans SecurityCenter. C'est aussi le premier groupe de données signalé par Nessus.
file_extension	Liste de toutes les extensions que Nessus doit rechercher. Les extensions sont répertoriées sans « . », entre guillemets, séparées par des pipes. Lorsque des options additionnelles telles que **regex** et **expect** ne sont pas incluses dans la vérification, les fichiers dotés de l'extension de fichier (file_extension) spécifiée sont affichés dans le rapport de sortie de vérification.
regex	Ce mot clé contient l'expression rationnelle utilisée pour rechercher des types complexes de données. Si l'expression rationnelle est correcte, le premier contenu correspondant est affiché dans le rapport des vulnérabilités. Le mot clé **regex** doit être exécuté avec le mot clé **expect** décrit ci-dessous. Contrairement aux contrôles de conformité Windows, les déclarations **regex** et **expect** des contrôles de conformité du contenu de fichier Windows n'ont pas besoin de correspondre à la même ou aux mêmes chaînes de données dans le fichier faisant l'objet de la recherche. Les contrôles de contenu de fichier Windows nécessitent seulement que les

	déclarations **regex** et **expect** correspondent aux données dans les octets <max_size> du fichier sur lequel la recherche est effectuée.
expect	La déclaration **expect** est utilisée pour répertorier un ou plusieurs motifs simples qui doivent figurer dans le document pour qu'il puisse correspondre. Par exemple, lors de la recherche des numéros de sécurité sociale, le mot « SSN », « SS# » ou « Social » pourrait être requis. Les motifs multiples sont répertoriés entre guillemets et séparés par des caractères de pipe. La correspondance des motifs simples est aussi prise en charge dans ce mot clé avec le point. Pour la correspondance de la chaîne « C.T », la déclaration **expect** correspondrait à « CAT », « CaT », « COT », « C T », etc. Le mot clé **expect** peut être exécuté de façon autonome pour la correspondance de motif simple, mais si le mot clé **regex** est utilisé, **expect** est requis. Contrairement aux contrôles de conformité Windows, les déclarations **regex** et **expect** des contrôles de conformité du contenu de fichier Windows n'ont pas besoin de correspondre à la même ou aux mêmes chaînes de données dans le fichier qui fait l'objet de la recherche. Les contrôles de contenu de fichier Windows nécessitent seulement que les déclarations **regex** et **expect** correspondent aux données dans les octets <max_size> du fichier sur lequel la recherche est effectuée.
file_name	Lorsque le mot clé **file_extension** est requis, ce mot clé peut affiner davantage la liste des fichiers à analyser. En fournissant une liste de motifs, les fichiers peuvent être rejetés ou assortis. Par exemple, ceci facilite beaucoup la recherche de tout type de nom de fichier dont le nom contient des expressions telles que « employé », « client » ou « salaire ».
max_size	Pour la mise en œuvre, un audit peut vouloir seulement analyser la première partie de chaque fichier. Ceci peut être spécifié en octets avec ce mot clé. Le nombre d'octets peut être utilisé comme argument. Les extensions « K » ou « M » pour kilooctets ou mégaoctets, respectivement, sont aussi prises en charge.
only_show	Lors de la correspondance de données sensibles telles que les numéros de carte de crédit, l'organisation peut exiger que seuls les quatre derniers chiffres soient visibles dans le rapport. Ce mot clé accepte la divulgation d'un nombre quelconque d'octets spécifié par la stratégie.
regex_replace	Ce mot clé contrôle le motif de l'expression rationnelle qui est montré dans le rapport. Lors de la recherche de motifs de données complexes telles que les numéros de carte de crédit, il n'est pas toujours possible que la première correspondance soit celle des données souhaitées. Ce mot clé fournit une meilleure polyvalence pour capturer les données souhaitées avec une plus grande précision.
include_paths	Ce mot clé permet d'inclure un répertoire ou une unité de lecture dans les résultats de la recherche. Il peut être utilisé de concert avec le mot clé « **exclude_paths** » ou indépendamment de ce dernier. Ceci est particulièrement utile lorsque seuls certains

	dossiers ou certaines unités de lecture doivent être recherchés sur un système à plusieurs unités de lecture. Les chemins sont indiqués avec des guillemets doubles et séparés par le symbole de pipe lorsque plusieurs chemins sont requis. Seuls les noms de dossier ou les lettres d'unité de lecture peuvent être spécifiés avec le mot clé « **include_paths** ». Les noms de fichiers ne peuvent pas être inclus dans la chaîne des valeurs « **include_paths** ».
exclude_paths	Ce mot clé permet d'exclure l'unité de lecture, le répertoire ou le fichier des résultats de recherche. Il peut être utilisé de concert avec le mot clé « **include_paths** » ou indépendamment de ce dernier. Ceci est particulièrement utile lorsqu'une unité de lecture, un répertoire ou un fichier particulier doit être exclu des résultats de la recherche. Les chemins sont indiqués avec des guillemets doubles et séparés par le symbole de pipe lorsque plusieurs chemins sont requis.
see_also	Ce mot clé permet d'inclure des liens à une référence. Exemple : `see_also: "https://benchmarks.cisecurity.org/tools2/linux/CIS_Redhat_Linux_5_Benchmark_v2.0.0.pdf"`
solution	Ce mot clé permet d'inclure un texte « Solution », le cas échéant. Exemple : `solution : "Remove this file, if its not required"`
reference	Ce mot clé permet d'inclure des références croisées dans le fichier **.audit**. Il respecte le format « ref|ref-id1,ref|ref-id2 ». Exemple : `reference : "CAT|CAT II,800-53|IA-5,8500.2|IAIA-1,8500.2|IAIA-2,8500.2|IATS-1,8500.2|IATS-2"`

Exemples de ligne de commande

Dans cette section, nous allons créer un document de texte fictif avec l'extension **.tns**, puis nous exécuterons plusieurs fichiers **.audit** simples ou complexes par rapport à ce document. À mesure que nous indiquerons chaque exemple, nous essaierons les différents cas des paramètres de Windows Content pris en charge.

Nous utiliserons aussi le binaire **nasl** de ligne de commande. Pour chacun des fichiers **.audit** que nous choisissons, vous pouvez facilement les analyser selon les stratégies de scan de Nessus 4 ou de SecurityCenter ; toutefois, pour les vérifications rapides d'un système, cette méthode est très efficace. La commande que nous exécuterons chaque fois à partir du répertoire **/opt/nessus/bin** est :

```
# ./nasl -t <IP>
      /opt/nessus/lib/nessus/plugins/compliance_check_windows_file_content.nbin
```

<IP> est l'adresse IP du système à vérifier.

Avec Nessus 4, lorsque **.nbin** (ou tout autre plugin) est exécuté, une invite demande les identifiants du système cible et l'emplacement du fichier **.audit**.

Fichier de test cible

Le fichier de test cible que nous utiliserons possède le contenu suivant :

```
abcdefghijklmnopqrstuvwxyz
01234567890
Tenable Network Security
SecurityCenter
Nessus
Passive Vulnerability Scanner
Log Correlation Engine
AB12CD34EF56
Nessus
```

Veuillez copier ces données dans tout système Windows pour lequel vous disposez d'un accès authentifié. Nommez le fichier « Tenable_Content.tns ».

Exemple 1 : Rechercher les documents `.tns` qui contiennent le mot « Nessus »

Voici un fichier **`.audit`** simple qui recherche les fichiers **`.tns`** contenant le mot « Nessus » n'importe où dans le document.

```
<check_type:"WindowsFiles">
<item>
 type: FILE_CONTENT_CHECK
 description: "TNS File that Contains the word Nessus"
 file_extension: "tns"
 expect: "Nessus"
</item>
</check_type>
```

Lorsque cette commande est exécutée, la sortie ci-dessous est attendue :

```
"TNS File that Contains the word Nessus" : [FAILED]
 - error message:
The following files do not match your policy :
Share: C$, path: \share\new folder\tenable_content.tns
```

Ces résultats montrent que nous avons trouvé un résultat. Le rapport déclare que nous avons « échoué » (failed) parce que nous avons trouvé des données que nous ne recherchions pas. Par exemple, si nous faisons un audit pour un numéro de sécurité sociale et que nous recevons un résultat positif de numéro de sécurité sociale sur l'ordinateur public, le résultat, bien qu'il soit positif, est enregistré en tant qu'échec pour des raisons de conformité.

Exemple 2 : Rechercher les documents `.tns` qui contiennent le mot « France »

Voici un fichier **`.audit`** qui recherche les fichiers **`.tns`** contenant le mot « France » n'importe où dans le document.

```
<check_type:"WindowsFiles">
<item>
 type: FILE_CONTENT_CHECK
 description: "TNS File that Contains the word France"
 file_extension: "tns"
 expect: "France"
</item>
</check_type>
```

Cette fois, nous obtenons la sortie suivante :

```
"TNS File that Contains the word France" : [PASSED]
```

Nous avons pu réussir la vérification parce qu'aucun des fichiers **.tns** que nous avons vérifiés ne contenait le mot « France ».

Exemple 3 : Rechercher les documents .tns et .doc qui contiennent le mot « Nessus »

L'ajout d'une deuxième extension pour la recherche des fichiers de documents de Microsoft Word est très facile, comme indiqué ci-dessous :

```
<check_type:"WindowsFiles">
<item>
 type: FILE_CONTENT_CHECK
 description: "TNS or DOC File that Contains the word Nessus"
 file_extension: "tns" | "doc"
 expect: "Nessus"
</item>
</check_type>
```

Les résultats (sur notre ordinateur de test) étaient les suivants :

```
"TNS or DOC File that Contains the word Nessus" : [FAILED]
 - error message:
The following files do not match your policy :
Share: C$, path: \share\new folder\tenable_content.tns
Share: C$, path: \documents and settings\jsmith\desktop\tns_roadmap.doc
```

Nous obtenons le même « échec » que précédemment avec notre fichier **.tns** de test, mais dans le cas présent il y avait un deuxième fichier, un fichier **.doc**, qui contenait aussi le mot « Nessus ». Si vous effectuez ces tests sur vos systèmes, vous pouvez avoir ou ne pas avoir de fichier Word contenant le mot « Nessus ».

Exemple 4 : Rechercher les documents .tns et .doc qui contiennent le mot « Nessus » et qui contiennent aussi un numéro à 11 chiffres

Nous allons maintenant ajouter notre première expression rationnelle pour rechercher un numéro à 11 chiffres. Il nous suffit d'ajouter l'expression rationnelle avec le mot clé **regex** au fichier **.audit** utilisé précédemment.

```
<check_type:"WindowsFiles">
<item>
 type: FILE_CONTENT_CHECK
 description: "TNS or DOC File that Contains the word Nessus"
 file_extension: "tns" | "doc"
 regex: " ([0-9]{11})"
 expect: "Nessus"
</item>
</check_type>
```

Cette exécution génère la sortie suivante :

```
"TNS or DOC File that Contains the word Nessus" : [FAILED]
 - error message:
The following files do not match your policy :
Share: C$, path: \share\new folder\tenable_content.tns (01234567890)
```

Le fichier **.doc** qui a été trouvé dans le dernier exemple fait toujours l'objet d'une recherche. Puisqu'il ne contient pas le numéro à 11 chiffres, il n'est plus affiché. En outre, il faut noter que, puisque nous utilisons le mot clé **regex**, nous obtenons aussi un résultat affiché dans les données.

Que se passerait-il si nous avions besoin de trouver un numéro à 10 chiffres ? Le numéro à 11 chiffres ci-dessus contient deux numéros à 10 chiffres (0123456789 et 1234567890). Si nous voulions écrire une correspondance plus précise pour 11 chiffres seulement, nous voudrions en fait une expression rationnelle qui déclarerait :

« Rechercher tout numéro à 11 chiffres qui n'est pas précédé ou suivi d'autres chiffres. »

Pour le faire dans les expressions rationnelles, nous pouvons ajouter l'opérateur « not » (non) comme suit :

```
<check_type:"WindowsFiles">
<item>
 type: FILE_CONTENT_CHECK
 description: "TNS or DOC File that Contains the word Nessus"
 file_extension: "tns" | "doc"
 regex: "([^0-9]|^)([0-9]{11})([^0-9]|$)"
 expect: "Nessus"
</item>
</check_type>
```

En lisant de gauche à droite, nous voyons aussi le caractère « ^ » et le symbole du dollar plusieurs fois. Le « ^ » indique parfois le début d'une ligne et il signifie parfois la correspondance d'un négatif. Le symbole du dollar signifie la fin d'une ligne. L'expression rationnelle ci-dessus signifie essentiellement de rechercher tout motif qui ne commence pas par un chiffre mais qui commence potentiellement sur une nouvelle ligne, contient 11 chiffres qui ne sont pas suivis par d'autres chiffres, et possède une fin de ligne. Les expressions rationnelles traitent le début et la fin d'une ligne comme des cas spéciaux, ce qui nécessite l'utilisation des caractères « ^ » ou « $ ».

Exemple 5 : Rechercher les documents .tns et .doc contenant le mot « Nessus » et un numéro à 11 chiffres, mais afficher seulement les 4 derniers octets

L'ajout du mot clé **only_show** à notre fichier **.audit** peut limiter la sortie. Ceci peut limiter les auditeurs à accéder uniquement aux données sensibles qu'ils recherchent.

```
<check_type:"WindowsFiles">
<item>
 type: FILE_CONTENT_CHECK
 description: "TNS or DOC File that Contains the word Nessus"
 file_extension: "tns" | "doc"
 regex: "([^0-9]|^)([0-9]{11})([^0-9]|$)"
 expect: "Nessus"
 only_show: "4"
</item>
</check_type>
```

Lorsqu'il y a correspondance, les données sont masquées par des caractères « X » comme indiqué ci-dessous :

```
"TNS or DOC File that Contains the word Nessus" : [FAILED]
 - error message:
The following files do not match your policy :
Share: C$, path: \share\new folder\tenable_content.tns (XXXXXXX7890)
```

Exemple 6 : Rechercher les documents .tns qui contiennent le mot « Correlation » dans les 50 premiers octets

Dans cet exemple, nous examinerons l'emploi du mot clé **max_size**. Dans notre fichier de test, le mot « Correlation » est situé à plus de 50 octets du commencement du fichier.

```
<check_type:"WindowsFiles">
<item>
 type: FILE_CONTENT_CHECK
 description: "TNS File that Contains the word Correlation"
 file_extension: "tns"
 expect: "Correlation"
 max_size: "50"
</item>
</check_type>
```

À l'exécution, nous obtenons une réussite :

```
"TNS File that Contains the word Correlation" : [PASSED]
```

Changez la valeur **max_size** « 50 » à « 50K » et exécutez à nouveau le scan. Nous obtenons alors une erreur :

```
"TNS File that Contains the word Correlation" : [FAILED]
 - error message:
The following files do not match your policy :
Share: C$, path: \share\new folder\tenable_content.tns
```

Exemple 7 : Contrôle de l'affichage en sortie

Dans cet exemple, nous examinerons l'emploi du mot **regex_replace**. Considérez le fichier **.audit** suivant :

```
<check_type:"WindowsFiles">
<item>
 type: FILE_CONTENT_CHECK
 description: "Seventh Example"
 file_extension: "tns"
 regex: "Passive Vulnerability Scanner"
 expect: "Nessus"
</item>
</check_type>
```

Ce contrôle produit la sortie suivante :

```
"Seventh Example" : [FAILED]
 - error message:
The following files do not match your policy :
Share: C$, path: \share\new folder\tenable_content.tns    (Passive Vulnerability
       Scanner)
```

Toutefois, vous pouvez considérer ce qui se passerait si nous avions vraiment besoin d'une expression rationnelle correspondant aux parties « Passive » (Passif) et « Scanner » (Scanner), mais si nous voulions renvoyer seulement la partie « Vulnerability ». Une nouvelle expression rationnelle ressemblerait à la suivante :

```
<check_type:"WindowsFiles">
<item>
 type: FILE_CONTENT_CHECK
 description: "Seventh Example"
 file_extension: "tns"
 regex: "(Passive) (Vulnerability) (Scanner)"
 expect: "Nessus"
</item>
</check_type>
```

Le contrôle produit toujours la correspondance complète de « Passive Vulnerability Scanner » parce que la déclaration de l'expression rationnelle traite l'ensemble de la chaîne comme la première correspondance. Pour obtenir seulement la deuxième correspondance, nous devons ajouter le mot clé **`regex_replace`**.

```
<check_type:"WindowsFiles">
<item>
 type: FILE_CONTENT_CHECK
 description: "Seventh Example"
 file_extension: "tns"
 regex: "(Passive) (Vulnerability) (Scanner)"
 regex_replace: "\3"
 expect: "Nessus"
</item>
</check_type>
```

La sortie du scan est la suivante :

```
"Seventh Example" : [FAILED]
 - error message:
The following files do not match your policy :
Share: C$, path: \share\new folder\tenable_content.tns    (Vulnerability)
```

Nous utilisons « \3 » pour indiquer le deuxième élément de correspondance parce que le premier (« \1 ») est la chaîne entière. Si nous avions utilisé « \2 », nous aurions produit le résultat « Passive » ; un « \4 » aurait produit « Scanner ».

Pourquoi cette fonction existe-t-elle ? Lors de la recherche de motifs de données complexes telles que les numéros de carte de crédit, il n'est pas toujours possible que la première correspondance soit celle des données souhaitées. Ce mot clé fournit une meilleure polyvalence pour capturer les données souhaitées avec une plus grande précision.

Exemple 8 : Utilisation du nom de fichier comme filtre

Le fichier **`.audit`** du troisième exemple a renvoyé un résultat pour un fichier **`.tns`** et un fichier **`.doc`**.

```
<check_type:"WindowsFiles">
<item>
 type: FILE_CONTENT_CHECK
 description: "TNS or DOC File that Contains the word Nessus"
 file_extension: "tns" | "doc"
 expect: "Nessus"
</item>
</check_type>
```

Les résultats (sur notre ordinateur de test) étaient les suivants :

```
"TNS or DOC File that Contains the word Nessus" : [FAILED]
 - error message:
The following files do not match your policy :
Share: C$, path: \share\new folder\tenable_content.tns
Share: C$, path: \documents and settings\jsmith\desktop\tns_roadmap.doc
```

Le mot clé **file_name** peut aussi être utilisé pour filtrer les fichiers que nous voulons ou que nous ne voulons pas. Pour l'ajouter au fichier **.audit** en indiquant de considérer seulement les fichiers dont le nom contient « tenable », procédez comme suit :

```
<check_type:"WindowsFiles">
<item>
 type: FILE_CONTENT_CHECK
 description: "TNS or DOC File that Contains the word Nessus"
 file_extension: "tns" | "doc"
 file_name: "tenable"
 expect: "Nessus"
</item>
</check_type>
```

La sortie est la suivante :

```
"TNS or DOC File that Contains the word Nessus" : [FAILED]
 - error message:
The following files do not match your policy :
Share: C$, path: \share\new folder\tenable_content.tns
```

Le fichier **.doc** résultant n'est pas présent parce que son chemin ne contenait pas le mot « tenable ».

La chaîne correspondante est une expression rationnelle, donc elle peut être très flexible pour faire correspondre une grande variété de fichiers que nous voulons ou que nous ne voulons pas. Par exemple, nous aurions pu utiliser la chaîne « [Tt]enable » pour obtenir le mot « Tenable » ou « tenable ». De même, si nous voulons obtenir comme résultat une extension ou une extension partielle, nous devons inclure le point dans une séquence d'échappement avec une barre oblique (« \.t ») pour rechercher toute extension commençant par « t ».

Exemple 9 : Utilisation des mots clés d'inclusion/exclusion

Les mots clés « **include_paths** » et « **exclude_paths** » peuvent être utilisés pour filtrer les recherches en fonction d'une exclusion de lettre d'unité de lecture, de répertoire et même de nom de fichier.

```
<item>
 type: FILE_CONTENT_CHECK
 description: "Does the file contain a valid VISA Credit Card Number"
 file_extension: "xls" | "pdf" | "txt"
 regex: "([^0-9-]|^)(4[0-9]{3}( |-|)([0-9]{4})( |-|)([0-9]{4})( |-|)([0-9]{4}))([^0-9-
       ]|$)"
 regex_replace: "\3"
 expect:"."
 max_size: "50K"
 only_show: "4"
 include_paths: "c:\" | "g:\" | "h:\"
 exclude_paths: "g:\dontscan"
</item>
```

La sortie est la suivante :

```
Windows File Contents Compliance Checks (Contrôles de conformité du contenu des
       fichiers Windows)
"Determine if a file contains a valid VISA Credit Card Number" : [FAILED]
- error message:
The following files do not match your policy :
Share: C$, path: \documents and settings\administrator\desktop\ccn.txt
       (XXXXXXXXXXXX0552)


Nessus ID : 24760
```

La sortie n'est pas différente d'un résultat standard de recherche du contenu de fichier Windows mais elle exclut le chemin exclu. Si un seul chemin est inclus à l'aide de « **include_paths** » (par exemple « **c:** »), tous les autres chemins sont exclus automatiquement. En outre, si une lettre d'unité de lecture est exclue (par exemple « **d:** ») mais qu'un dossier appartenant à cette unité est inclus (par exemple « **d:\users** »), le mot clé « **exclude_paths** » a priorité et la recherche exclura l'unité de lecture. Cependant, vous pouvez inclure une unité **C:** puis exclure un sous-dossier dans l'unité (par exemple, **C:\users:**).

Audit de types différents de formats de fichier

Toute extension de fichier peut être vérifiée ; toutefois, les fichiers tels que **.zip** et **.gz** ne sont pas décompressés sur-le-champ. Si le fichier inclut une compression ou un autre type de codage dans les données, la recherche des motifs peut ne pas être possible.

Pour les documents qui mémorisent les données au format Unicode, les programmes d'analyse du fichier **.nbin** élimineront tous les octets « NULL » trouvés.

En outre, toutes les versions des documents Microsoft Office sont prises en charge. Ceci inclut les versions codées plus récentes, ajoutées dans Office 2007, telles que **.xlsx** et **.docx**.

Enfin, la prise en charge de divers types de formats de fichier PDF est incluse. Tenable a écrit un analyseur PDF avancé qui extrait les chaînes brutes pour leur correspondance. Les utilisateurs ont seulement besoin de se préoccuper du type de données qu'ils souhaitent rechercher dans un fichier PDF.

Considérations de performance

Il existe plusieurs compromis que toute organisation doit considérer pour modifier les fichiers **.audit** par défaut et les tester sur des réseaux existants :

- Quelles sont les extensions à chercher ?
- Combien de données doivent être scannées ?

Les fichiers **.audit** ne nécessitent pas le mot clé **max_size**. Dans ce cas, Nessus essaie de récupérer le fichier complet et continue l'opération sauf s'il trouve une correspondance de motif. Puisque ces fichiers traversent le réseau, il existe davantage de trafic réseau avec ces vérifications qu'avec le scan ou les vérifications de configuration typiques.

Si plusieurs scanners Nessus sont gérés par un SecurityCenter, il suffit que les données soient transmises depuis l'hôte Windows scanné au scanner effectuant la vérification des vulnérabilités.

Référence des fichiers de conformité d'audit pour la configuration de Cisco IOS

Cette section décrit le format et les fonctions des contrôles de conformité Cisco IOS et la fonction de chaque paramètre.

Emploi des guillemets :

Les guillemets simples et doubles sont interchangeables autour des champs d'audit, sauf dans les deux cas suivants :

1. Dans les contrôles de conformité Windows où des champs spéciaux tels que CRLF, doivent être interprétés littéralement, utilisez des guillemets simples. Tout champ intégré qui doit être interprété comme une chaîne doit être inclus dans une séquence d'échappement.

Par exemple :
```
expect: 'First line\r\nSecond line\r\nJohn\'s Line'
```

2. Des guillemets doubles sont requis lorsque vous utilisez les fichiers Windows « include_paths » et « exclude_paths ».

Si des chaînes sont utilisées dans tout type de champ (description, value_data, regex, etc.) contenant des guillemets simples ou doubles, il y a deux façons de les traiter :

a. Utilisez le type de guillemets opposés pour les guillemets extérieurs.

Par exemple :
```
expect: "This is John's Line"
expect: 'We are looking for a double-quote-".*'
```

b. Échappez tout guillemet intégré avec une barre oblique inverse (guillemets doubles uniquement).

Par exemple :
```
expect: "\"Text to be searched\""
```

Type de contrôle

Tous les contrôles de conformité Cisco IOS doivent être mis entre crochets avec l'encapsulation **`check_type`** et la désignation « Cisco ». Ceci est requis pour distinguer les fichiers **`.audit`** spécifiquement conçus pour les systèmes exécutant le système d'exploitation Cisco IOS des autres types d'audit de conformité.

Exemple :

```
<check_type:"Cisco">
```

Contrairement à d'autres types d'audit de conformité, aucun mot clé supplémentaire de type ou de version n'est disponible.

Mots clés

Le tableau suivant indique comment chaque mot clé peut être utilisé dans les contrôles de conformité de Cisco :

Mot clé	Exemple d'utilisation et paramètres pris en charge
`type`	CONFIG_CHECK, CONFIG_CHECK_NOT et RANDOMNESS_CHECK « CONFIG_CHECK » (contrôle de configuration) détermine si l'élément spécifié existe dans la sortie « show config » (montrer config.) de CISCO IOS. De la même façon, « CONFIG_CHECK_NOT » (contrôle de non configuration) détermine si l'élément spécifié n'existe pas. « RANDOMNESS_CHECK » (contrôle du caractère aléatoire) est utilisé pour effectuer des contrôles de complexité de chaîne (par exemple, contrôles de mot de passe). Si un élément à rechercher est spécifié (par une regex), le résultat indique si la chaîne est suffisamment « aléatoire » (au moins huit caractères de long,

	avec majuscule, minuscule, au moins un chiffre et au moins un caractère spécial).  Les paramètres aléatoires ne sont pas configurables pour le moment.
description	Le mot clé « **description** » permet d'ajouter une brève description du contrôle en cours. Il est fortement recommandé que le champ **description** soit unique et qu'aucun autre contrôle ne possède le même champ **description**. SecurityCenter de Tenable utilise ce champ pour produire automatiquement un numéro d'ID de plugin unique basé sur le champ **description**. Exemple : `description: "Forbid Remote Startup Configuration"`

feature_set	Le mot clé « **feature_set** », similaire au mot clé « **system** » dans les contrôles de conformité Unix, vérifie la version de Feature Set (Ensemble de fonctions) de Cisco IOS et renvoie la vérification résultante ou omet la vérification à cause d'un échec de regex. Ceci est utile pour les cas où un contrôle est applicable uniquement aux systèmes qui possèdent un ensemble de fonctions particulier.
	
	Exemple :
	`<item>`
	` type: CONFIG_CHECK`
	` description: "Version Check"`
	` info: "SSH Access Control Check."`
	` feature_set: "K8" context:"line .*"`
	` item: "access-class [0-9]+ in"`
	`</item>`
	
	Le contrôle ci-dessus exécute seulement « item » si la version de l'ensemble de fonctions correspond à la regex spécifiée : (K8)
	
	En cas d'échec du contrôle de la version de l'ensemble de fonctions, une erreur similaire à l'erreur ci-dessous est affichée :
	
	`"Version Check" : [SKIPPED]`
	`Test defined for the feature set 'K8' whereas we are running C850-ADVSECURITYK9-M`

ios_version	Le mot clé « **ios_version** », similaire au mot clé « **system** » dans les contrôles de conformité Unix, vérifie la version de Cisco IOS et renvoie le contrôle résultant ou omet le contrôle à cause d'un échec de regex. Ceci est utile pour les cas où un contrôle est applicable uniquement aux systèmes avec une version IOS particulière.
	
	Exemple :
	`<item>`
	` type: CONFIG_CHECK`
	` description: "Version Check"`
	` info: "SSH Access Control Check."`
	` ios_version: "12\.[5-9]"`
	` context: "line .*"`
	` item: "access-class [0-9]+ in"`
	`</item>`
	
	Le contrôle ci-dessus exécute seulement « item » si la version de l'ensemble de

	fonctions correspond à la regex spécifiée : (12\.[5-9]).
	
	En cas d'échec du contrôle de la version IOS, une erreur similaire à l'erreur ci-dessous est affichée :
	
	`"Version Check" : [SKIPPED]`
	`Test defined for 12.[5-9] whereas we are running 12.4(15)T10`

info	Le mot clé « **info** » est utilisé pour ajouter une description plus détaillée au contrôle en cours. La raison de ce contrôle pourrait être une réglementation, une URL avec plus d'informations, une stratégie d'entreprise et plus. Plusieurs champs **info** peuvent être ajoutés sur des lignes séparées pour formater le texte sous forme de paragraphe. Il n'existe pas de limite prédéfinie pour le nombre de champs **info** qui peuvent être utilisés.
	
	  Chaque balise « **info** » doit être écrite sur une ligne séparée sans saut de ligne. Si plusieurs lignes sont requises (par exemple, pour des raisons de formatage), ajoutez des balises « **info** » supplémentaires.
	
	Exemple :
	`info: "Verify at least one local user exists and ensure"`
	`info: "all locally defined user passwords are protected"`
	`info: "by encryption."`

item	Le mot clé « **item** » spécifie l'élément de configuration dans la sortie du résultat de « show config » à vérifier. Exemple : `item: "transport input ssh"` Des expressions rationnelles peuvent être utilisées dans ce mot clé pour filtrer les résultats du contrôle. Voir la description du mot clé **regex** pour plus d'informations concernant la fonctionnalité **regex**.
regex	Le mot clé « **regex** » permet de chercher le paramètre d'élément de configuration pour trouver une expression rationnelle particulière. Exemple : `regex: "snmp-server community ([^ ]*) .*"` Les caractères génériques suivants nécessitent un traitement spécial : + \ * ( ) ^ Échapper deux fois ces caractères avec deux barres obliques inverses « \\ » ou les placer entre crochets « [] » s'ils doivent être interprétés littéralement. D'autres caractères tels que ceux indiqués ci-dessous nécessitent seulement une barre oblique inverse pour être interprétés littéralement : . ? " ' La raison provient de la façon dont le compilateur traite ces caractères.
min_occurrences	Le mot clé « **min_occurrences** » spécifie le nombre minimum requis de détections de l'élément de configuration pour assurer la réussite du contrôle. Exemple : `min_occurrences: "3"`
max_occurrences	Le mot clé « **max_occurrences** » spécifie le nombre maximum autorisé de

	détections de l'élément de configuration pour assurer la réussite du contrôle. Exemple : `max_occurrences: "1"`
required	Le mot clé « **required** » est utilisé pour spécifier si l'élément vérifié doit être présent sur le système distant. Par exemple, si **required** est paramétré sur « NO » (Non) et que le type de contrôle est « CONFIG_CHECK », le contrôle réussit si l'élément de configuration existe ou si l'élément de configuration n'existe pas. D'un autre côté, si **required** est paramétré sur « YES » (Oui), le contrôle ci-dessus échoue. Exemple : `required: NO`

context	Le mot clé « **context** » est utile lorsqu'il existe plus d'un exemplaire d'un élément de configuration particulier. Par exemple, considérez la configuration suivante :
	
	`line con 0`
	` no modem enable`
	`line aux 0`
	` access-class 42 in`
	` exec-timeout 10 0`
	` no exec`
	`line vty 0 4`
	` exec-timeout 2 0`
	` password 7 15010X1C142222362G`
	` transport input ssh`
	
	Pour tester une valeur sur une ligne série particulière, l'emploi du mot clé **item** avec « line » (ligne) n'est pas suffisant car il existe plus d'une option de « line ». Si vous utilisez « **context** », vous vous concentrez seulement sur l'élément qui vous intéresse. Par exemple :
	
	`context: "con 0"`
	
	Vous effectuez seulement une recherche de l'élément de configuration suivant :
	
	`line con 0`
	` no modem enable`
	
	Des expressions rationnelles peuvent être utilisées dans ce mot clé pour filtrer les résultats du contrôle. Voir la description du mot clé **regex** pour plus d'informations concernant la fonctionnalité **regex**.

Exemples de ligne de commande

Cette section fournit quelques exemples de vérifications fréquemment utilisées pour les contrôles de conformité Cisco IOS. Le binaire de ligne de commande **nasl** est utilisé comme moyen rapide de tester les audits sur-le-champ. Chacun des fichiers **.audit** démontrés ci-dessus peut être facilement importé dans les stratégies de scan Nessus. Toutefois, pour les audits rapides d'un système, les tests par ligne de commande sont plus efficaces. La commande sera exécutée chaque fois à partir du répertoire **/opt/nessus/bin** comme suit :

```
# ./nasl -t <IP> /opt/nessus/lib/nessus/plugins/cisco_compliance_check.nbin
```

<IP> est l'adresse IP du système à vérifier.

Le mot de passe « enable » (activer) est demandé :

```
Which file contains your security policy ? cisco_test.audit
SSH login to connect with : admin
How do you want to authenticate ? (key or password) [password]
SSH password :
Enter the 'enable' password to use :
```

Consultez l'administrateur Cisco pour les paramètres corrects de connexion « enable ».

Exemple 1 : Rechercher une ACL SNMP définie

Le fichier **`.audit`** simple indiqué ci-dessous recherche une ACL SNMP « deny » définie. Si aucune n'est trouvée, l'audit affiche un message d'échec. Cette vérification est exécutée seulement si la version IOS du routeur correspond à la regex spécifiée. Sinon, le contrôle sera omis.

```
<check_type: "Cisco">

<item>
 type: CONFIG_CHECK
 description: "Require a Defined SNMP ACL"
 info: "Verify a defined simple network management protocol (SNMP) access control list
       (ACL) exists with rules for restricting SNMP access to the device."
 ios_version: "12\.[4-9]"
 item: "deny ip any any"
</item>

</check_type>
```

Lors de l'exécution de cette commande, la sortie suivante est attendue pour un système conforme :

```
"Require a Defined SNMP ACL" : [PASSED]

Verify a defined simple network management protocol (SNMP) access control list (ACL)
       exists with rules for restricting SNMP access to the device.
```

Un échec de l'audit renverrait la sortie suivante :

```
"Require a Defined SNMP ACL" : [FAILED]

Verify a defined simple network management protocol (SNMP) access control list (ACL)
       exists with rules for restricting SNMP access to the device.

 - error message: deny ip any any not found in the configuration file
```

Dans ce cas, le contrôle a échoué parce que nous recherchions une règle « deny ip » et qu'aucune n'a été trouvée.

Exemple 2 : S'assurer que le service « `finger` » est désactivé

Ce fichier **`.audit`** simple recherche le service non sécurisé « **`finger`** » sur le routeur distant. Cette vérification est exécutée seulement si la version IOS du routeur correspond à la regex spécifiée. Sinon, le contrôle sera omis. Si le service est trouvé, l'audit affiche un message d'échec.

```
<check_type: "Cisco">

<item>
```

```
 type: CONFIG_CHECK_NOT
 description: "Forbid Finger Service"
 ios_version: "12\.[4-9]"
 info: "Disable finger server."
 item: "(ip|service) finger"
</item>

</check_type>
```

Lors de l'exécution de cette commande, la sortie suivante est attendue pour un système conforme :

```
"Forbid Finger Service" : [PASSED]

Disable finger server.
```

Un échec de l'audit retournerait la sortie suivante :

```
"Forbid Finger Service" : [FAILED]
Disable finger server.
 - error message:
The following configuration line is set:
ip finger <----

Policy value:
(ip|service) finger
```

Exemple 3 : Contrôle de caractère aléatoire pour vérifier que les chaînes de communauté SNMP et le contrôle d'accès sont suffisamment aléatoires

Le fichier **`.audit`** simple ci-dessous recherche des chaînes de communauté SNMP qui sont insuffisamment aléatoires. Si une chaîne de communauté qui n'est pas déterminée comme étant suffisamment aléatoire est découverte, l'audit affiche un message d'erreur. Puisque l'option « required » (obligatoire) est paramétrée sur « NO » (Non), la vérification réussit s'il n'existe aucune chaîne de communauté snmp-server. Cette vérification sera exécutée seulement si le routeur utilise un ensemble de fonctions « K9 ». Sinon, le contrôle sera omis.

```
<check_type: "Cisco">

<item>
 type: RANDOMNESS_CHECK
 description: "Require Authorized Read SNMP Community Strings and Access Control"
 info: "Verify an authorized community string and access control is configured to
       restrict read access to the device."
 feature_set: "K9"
 regex: "snmp-server community ([^ ]*) .*"
 required: NO
</item>

</check_type>
```

Lors de l'exécution de cette commande, la sortie suivante est attendue pour un système conforme :

```
"Require Authorized Read SNMP Community Strings and Access Control" : [PASSED]
```

```
Verify an authorized community string and access control is configured to restrict
       read access to the device.
```

Un échec de l'audit renverrait la sortie suivante :

```
"Require Authorized Read SNMP Community Strings and Access Control" : [FAILED]

Verify an authorized community string and access control is configured to restrict
       read access to the device.
 - error message:

The following configuration line does not contain a token deemed random enough:
snmp-server community foobar RO

The following configuration line does not contain a token deemed random enough:
snmp-server community public RO
```

Dans le cas ci-dessus, les deux chaînes « foobar » et « public » n'avaient pas d'unité lexicale suffisamment aléatoire et elles ont donc échoué au contrôle.

Exemple 4 : Contrôle de contexte pour vérifier le contrôle d'accès SSH

Le fichier **`.audit`** simple ci-dessous recherche tous les éléments de configuration « line » à l'aide du mot clé « **`context`** » et effectue une **`regex`** pour déterminer si le contrôle d'accès SSH est paramétré.

```
<check_type: "Cisco">

<item>
 type: CONFIG_CHECK
 description: "Require SSH Access Control"
 info: "Verify that management access to the device is restricted on all VTY lines."
 context: "line .*"
 item: "access-class [0-9]+ in"</item>
</item>

</check_type>
```

Lors de l'exécution de cette commande, la sortie suivante est attendue pour un système conforme :

```
"Require SSH Access Control" : [PASSED]

 Verify that management access to the device is restricted on all VTY lines.
```

Un échec de l'audit renverrait la sortie suivante :

```
"Require SSH Access Control" : [FAILED]

 Verify that management access to the device is restricted on all VTY lines.

 - error message:
The following configuration is set:
line con 0
```

```
 exec-timeout 5 0
 no modem enable

Missing configuration: access-class [0-9]+ in

The following configuration is set:
line vty 0 4
 exec-timeout 5 0
 password 7 15010A1C142222362D
 transport input ssh

Missing configuration: access-class [0-9]+ in
```

Dans le cas ci-dessus, deux chaînes correspondaient à la regex de mot clé « **context** » de « **line .*** ». Comme aucune ligne ne contenait la regex « **item** », la vérification a renvoyé un message « FAILED » (Échec).

Conditions

Il est possible de définir la logique **if**/**then**/**else** dans la stratégie d'audit Cisco. Ceci permet à l'utilisateur final de renvoyer un message d'avertissement plutôt qu'un message de réussite/échec en cas de réussite de l'audit.

La syntaxe pour exécuter les conditions est la suivante :

```
<if>
 <condition type: "or">
  <Insert your audit here>
 </condition>
 <then>
  <Insert your audit here>
 </then>
 <else>
  <Insert your audit here>
 </else>
</if>
```

Exemple :

```
<if>
<condition type: "AND">
 <item>
  type: CONFIG_CHECK
  description: "Forbid Auxiliary Port"
  info: "Verify the EXEC process is disabled on the auxiliary (aux) port."
  context: "line aux "
  item: "no exec"
 </item>
 <item>
  type: CONFIG_CHECK_NOT
  description: "Forbid Auxiliary Port"
  info: "Verify the EXEC process is disabled on the auxiliary (aux) port."
  context: "line aux "
  item: "transport input [^n][^o]?[^n]?[^e]?$"
 </item>
</condition>
<then>
```

```
 <report type: "PASSED">
  description: "Forbid Auxiliary Port"
  info: "Verify the EXEC process is disabled on the auxiliary (aux) port."
 </report>
</then>
<else>
 <report type: "FAILED">
  description: "Forbid Auxiliary Port"
  info: "Verify the EXEC process is disabled on the auxiliary (aux) port."
</report>
</else>
</if>
```

L'échec ou la réussite de la condition n'est jamais indiqué dans le rapport car il s'agit d'une vérification « muette ».

Les conditions peuvent être du type « **and** » ou « **or** ».

Référence des fichiers de conformité d'audit pour la configuration Juniper

Cette section décrit le format et les fonctions de contrôles de conformité Juniper, ainsi que les raisons de chaque paramètre.

Emploi des guillemets :

Les guillemets simples et doubles sont interchangeables autour des champs d'audit, sauf dans les deux cas suivants :

1. Dans les contrôles de conformité Windows où des champs spéciaux tels que CRLF, doivent être interprétés littéralement, utilisez des guillemets simples. Tout champ intégré qui doit être interprété comme une chaîne doit être inclus dans une séquence d'échappement.

Par exemple :
```
expect: 'First line\r\nSecond line\r\nJohn\'s Line'
```

2. Des guillemets doubles sont requis lorsque vous utilisez les fichiers Windows « include_paths » et « exclude_paths ».

Si des chaînes sont utilisées dans tout type de champ (description, value_data, regex, etc.) contenant des guillemets simples ou doubles, il y a deux façons de les traiter :

a. Utilisez le type de guillemets opposés pour les guillemets extérieurs.

Par exemple :
```
expect: "This is John's Line"
expect: 'We are looking for a double-quote-".*'
```

b. Échappez tout guillemet intégré avec une barre oblique inverse (guillemets doubles uniquement).

Par exemple :
```
expect: "\"Text to be searched\""
```

Check Type: CONFIG_CHECK

Les contrôles de conformité Juniper sont placés entre crochets dans l'encapsulation **custom_item** et CONFIG_CHECK ou SHOW_CONFIG_CHECK. Ils sont traités comme les autres fichiers **.audit** et ils s'appliquent aux systèmes qui utilisent le système d'exploitation Juniper (Junos). La vérification « CONFIG_CHECK » comprend au moins deux mots

clés. Les mots clés **type** et **description** sont obligatoires et ils sont suivis d'un ou de plusieurs mots clés. Le contrôle vérifie la configuration dans le format « set ».

La configuration dans le format « set » peut être obtenue en ajoutant « display set » à la demande « show configuration ». Par exemple :

```
show configuration | display set
```

```
admin> show configuration | display set
set version 10.2R3.10
set system time-zone GMT
set system no-ping-record-route
set system root-authentication encrypted-password "$1$hSGSlnwfdsdfdfsdfsdf43534"
```

Mots clés

Le tableau suivant indique comment chaque mot clé peut être utilisé dans les contrôles de conformité Juniper :

Mot clé	Exemple d'utilisation et paramètres pris en charge
type	CHECK_CONFIG et SHOW_CHECK_CONFIG « CHECK_CONFIG » détermine si l'élément de config spécifié existe dans la sortie « show configuration » de Juniper au format « set ». De même, « SHOW_CONFIG_CHECK » vérifie si l'élément de config spécifié existe dans la sortie « show configuration » au format par défaut.
description	Le mot clé « **description** » permet d'ajouter une brève description du contrôle en cours. Il est fortement recommandé que le champ **description** soit unique et qu'aucun autre contrôle ne possède le même champ **description**. SecurityCenter de Tenable utilise ce champ pour produire automatiquement un numéro d'ID de plugin unique basé sur le champ **description**. Exemple : `description: " 3.1 Disable Unused Interfaces"`

info	Le mot clé « **info** » est utilisé pour ajouter une description plus détaillée au contrôle en cours. La raison de ce contrôle pourrait notamment être une réglementation, une URL avec plus d'informations ou une stratégie d'entreprise. Plusieurs champs **info** peuvent être ajoutés sur des lignes séparées pour formater le texte sous forme de paragraphe. Il n'existe pas de limite prédéfinie pour le nombre de champs **info** qui peuvent être utilisés.
	
	Chaque balise « **info** » doit être écrite sur une ligne séparée sans saut de ligne. Si plusieurs lignes sont requises (par exemple, pour des raisons de formatage), ajoutez des balises « **info** » supplémentaires.
	
	Exemple :
	`info: "Review the the list of interfaces"`
	`info: "Disable unused interfaces"`

severity	Le mot clé « **severity** » spécifie la gravité du contrôle en cours.

	Exemple : `severity: MEDIUM` La gravité peut être paramétrée sur HIGH (Élevée), MEDIUM (Moyenne) ou LOW (Faible).
regex	Le mot clé « **regex** » permet de chercher le paramètre d'élément de configuration pour trouver une expression rationnelle particulière. Exemple : `regex: " set system syslog .+"` Les caractères génériques suivants nécessitent un traitement spécial : + \ * ( ) ^ Échapper deux fois ces caractères avec deux barres obliques inverses « \\ » ou les placer entre crochets « [] » s'ils doivent être interprétés littéralement. D'autres caractères tels que ceux indiqués ci-dessous nécessitent seulement une barre oblique inverse pour être interprétés littéralement : . ? " ' La raison provient de la façon dont le compilateur traite ces caractères. Si une balise « **regex** » est définie pour un contrôle, mais pas une balise « **expect** », « **not_expect** » ou « **number_of_lines** », le contrôle signale simplement toutes les lignes correspondant au mot clé regex.

expect	Ce mot clé permet de vérifier l'élément de configuration qui correspond à la balise « **regex** ». Si la balise « **regex** » n'est pas utilisée, il recherche la chaîne « **expect** » dans l'ensemble de config.
	
	Exemple :
	`expect: "syslog host 1.1.1.1"`
	
	Le contrôle renvoie un résultat concluant lorsque la ligne de config trouvée par « **regex** » correspond à la balise « **expect** » ou, si « **regex** » n'est pas défini, lorsque la chaîne « **expect** » est trouvée dans config.
	
	Exemple :
	`regex: "syslog host [0-9\.]+"`
	`expect: "syslog host 1.1.1.1"`
	
	Dans le cas ci-dessus, la balise « **expect** » garantit que l'hôte syslog est paramétré sur 1.1.1.1.

not_expect	Ce mot clé permet de rechercher les éléments de configuration qui ne devraient pas figurer dans la configuration. Exemple : `not_expect: "syslog host 1.1.1.1"` Il a l'action contraire de « **expect** ». Le contrôle est concluant lorsque la ligne de config trouvée par « **regex** » ne correspond pas à la balise « **not_expect** » ou, si « **regex** » n'est pas défini, lorsque la chaîne « **not_expect** » n'est pas trouvée dans config.

	Exemple :
	`regex: "syslog host [0-9\.]+"`
	`not_expect: "syslog host 1.1.1.1"`
	
	Dans le cas ci-dessus, la balise « **not_expect** » garantit que l'hôte syslog n'est pas paramétré sur 1.1.1.1.

number_of_lines	Ce mot clé permet de tester la conformité d'un contrôle d'audit en fonction du nombre de lignes correspondantes retournées par config.
	
	`<custom_item>`
	`type: CONFIG_CHECK`
	`description: "Syslog"`
	`regex: "syslog host [0-9\.]+"`
	`number_of_lines: "^1$"`
	`</custom_item>`
	
	Dans le cas ci-dessus, le contrôle indique une réussite lorsqu'une seule ligne correspondant à « **regex** » est renvoyée.

Exemples de CONFIG_CHECK

Les exemples qui suivent illustrent l'utilisation de CONFIG_CHECK pour un périphérique Juniper :

```
<custom_item>
 type: CONFIG_CHECK
 description: "Audit Syslog host message severity"
 regex: "syslog host [0-9\.]+"
 expect: "syslog host [0-9\.]+ 6 .+"
</custom_item>
```

```
<custom_item>
 type: CONFIG_CHECK
 description: "Audit Syslog host"
 regex: "syslog host [0-9\.]+"
 number_of_lines: "^1$"
</custom_item>
```

```
<custom_item>
 type: CONFIG_CHECK
 description: "Audit Syslog host"
 regex: "syslog host [0-9\.]+"
 not_expect: "syslog host 1.2.3.4"
</custom_item>
```

```
<custom_item>
 type: CONFIG_CHECK
 description: "Audit Syslog settings"
 regex: "syslog .+"
</custom_item>
```

Check Type: SHOW_CONFIG_CHECK

Ce contrôle vérifie à bien des égards les mêmes paramètres que le contrôle .audit CONFIG_CHECK. Cependant, le format de la configuration vérifiée est différent. SHOW_CONFIG_CHECK vérifie la configuration dans son format par défaut.

Par exemple, voici la configuration au format par défaut :

```
admin> show configuration system syslog
user * {
    any emergency;
}
host 1.1.1.1 {
    any none;
}
file messages {
    any any;
    authorization info;
}
file interactive-commands {
    interactive-commands any;
}
```

Ce contrôle n'est pas recommandé sauf si vous recherchez une plus grande flexibilité par rapport à CONFIG_CHECK. Puisque chaque contrôle .audit SHOW_CONFIG_CHECK entraîne l'exécution d'une commande séparée sur le périphérique Juniper, le processus peut entraîner un temps processeur supplémentaire et allonger la durée d'exécution. Ce contrôle a pour but de fournir une flexibilité à l'auditeur et de prendre en charge un cas d'utilisation futur éventuel pour lequel une vérification avec CONFIG_CHECK ne serait pas efficace.

Mots clés

Le tableau suivant indique comment chaque mot clé peut être utilisé dans les contrôles de conformité Junos. Veuillez noter que la conformité d'un contrôle peut être déterminée en comparant la sortie du contrôle à la balise « **expect** », « **not_expect** » ou « **number_of_lines** ». Il ne peut y avoir qu'une seule balise de test de conformité (« **expect** », « **not_expect** » ou « **number_of_lines** », mais pas « **expect** » et « **not_expect** »).

Mot clé	Exemple d'utilisation et paramètres pris en charge

hierarchy	Ce mot clé permet aux utilisateurs de naviguer jusqu'à une hiérarchie spécifique dans la configuration Junos.
	
	Exemple :
	`hierarchy: "interfaces"`
	
	À l'interne, le mot clé hierarchy est ajouté à la commande « show configuration » dans SHOW_CONFIG_CHECK. Par exemple :
	
	`<custom_item>`
	` type: SHOW_CONFIG_CHECK`
	` description: "3.6 Forbid Multiple Loopback Addresses"`
	` hierarchy: "interfaces"`
	`</custom_item>`
	
	Le contrôle ci-dessus revient à exécuter :

	`show configuration interfaces`

property	Ce mot clé permet aux utilisateurs de vérifier une propriété (« **property** ») spécifique sur le périphérique Junos. Par défaut, le contrôle SHOW_CONFIG_CHECK vérifie la commande « show configuration », suivie d'au moins un mot clé tel que **match**, **except** ou **find**. Dans le cas où le mot clé « **property** » est paramétré, il vérifie la propriété spécifique.
	
	Exemple :
	`property: "ospf"`
	
	`<custom_item>`
	` type: SHOW_CONFIG_CHECK`
	` description: "4.3.1 Require MD5 Neighbor Authentication (where OSPF is used)"`
	` info: "Level 2, Scorable"`
	` property: "ospf"`
	` hierarchy: "interface detail"`
	` match: "Auth type MD5"`
	`</custom_item>`
	
	Le contrôle ci-dessus revient à exécuter :
	
	`show ospf interface detail`
	
	Veillez noter que l'exemple ci-dessus n'a pas exécuté « show configuration », comme dans les autres exemples.

find	Ce mot clé recherche la hiérarchie de config appropriée dans un contrôle **.audit** SHOW_CONFIG_CHECK.
	
	`find: "chap"`
	
	Le mot clé **find** est ajouté à la demande « show configuration ».
	
	`<custom_item>`
	` type: SHOW_CONFIG_CHECK`
	` description: "3.8.2 Require CHAP Authentication if Incoming Map is Used"`
	` hierarchy: "interfaces"`
	` find: "chap"`
	` match: "access-profile"`
	`</custom_item>`
	
	Le contrôle ci-dessus revient à exécuter :
	
	`show configuration interfaces | find "chap" | match "access-profile"`

match	Ce mot clé recherche les lignes correspondantes dans un contrôle .audit SHOW_CONFIG_CHECK.
	
	`match: "multihop"`
	
	Le mot clé **match** est ajouté à la demande « show configuration ».
	
	`<custom_item>`
	` type: SHOW_CONFIG_CHECK`
	` description: "3.6 Forbid Multiple Loopback Addresses"`
	` hierarchy: "interfaces"`
	` match: "lo[0-9]"`
	`</custom_item>`
	
	Le contrôle ci-dessus revient à exécuter :
	
	`show configuration interfaces | match "lo[0-9]"`

except	Ce mot clé exclut certaines lignes de config dans un contrôle .audit SHOW_CONFIG_CHECK.
	
	`except: "multihop"`
	
	Le mot clé **except** est ajouté à la demande « show configuration ».
	
	`<custom_item>`
	` type: SHOW_CONFIG_CHECK`
	` description: "6.8.1 Require External Time Sources"`
	` hierarchy: "system ntp"`
	` match: "server"`
	` except: "boot-server"`
	`</custom_item>`
	
	Le contrôle ci-dessus revient à exécuter :
	
	`show configuration system ntp | match "server" | except "boot-server"`

expect	Ce mot clé permet de vérifier l'élément de config qui correspond à la balise « **regex** ». Si la balise « **regex** » n'est pas utilisée, il recherche la chaîne « **expect** » dans l'ensemble de config. Le contrôle renvoie un résultat concluant lorsque la ligne de config trouvée par « **regex** » correspond à la balise « **expect** » ou, si « **regex** » n'est pas défini, lorsque la chaîne « **expect** » est trouvée dans config.
	
	`regex: "syslog host [0-9\.]+"`
	`expect: "syslog host 1.2.4.5"`

	Dans le cas ci-dessus, la balise « **expect** » garantit que la complexité est définie avec une valeur comprise entre 1 et 4. `expect: "syslog host"` Dans le cas ci-dessus, la balise « **expect** » garantit que la valeur 4 est attribuée à la complexité.

not_expect	Ce mot clé permet de rechercher les éléments de configuration qui ne devraient pas figurer dans la configuration.
	
	Il a l'action contraire de « **expect** ». Le contrôle est concluant lorsque la ligne de config trouvée par « **regex** » ne correspond pas à la balise « **not_expect** » ou, si « **regex** » n'est pas défini, lorsque la chaîne « **not_expect** » n'est pas trouvée dans config.
	
	`regex: "syslog host [0-9\.]+"`
	`not_expect: "syslog host 1.2.3.4"`
	
	`not_expect: "syslog host"`

number_of_lines	Ce mot clé permet de tester la conformité d'un contrôle .audit en fonction du nombre de lignes correspondantes renvoyées par config.
	
	`<custom_item>`
	` type: CONFIG_CHECK`
	` description: "Syslog"`
	` regex: "syslog host [0-9\.]+"`
	` number_of_lines: "^1$"`
	`</custom_item>`
	
	Dans le cas ci-dessus, le contrôle indique une réussite lorsqu'une seule ligne correspondant à « **regex** » est renvoyée.

Exemples de SHOW_CONFIG_CHECK

Les exemples qui suivent illustrent l'utilisation de SHOW_CONFIG_CHECK pour un périphérique Juniper :

```
<custom_item>
 type: SHOW_CONFIG_CHECK
 description: "6.1.2 Require Accounting of Logins & Configuration Changes"
 hierarchy: "system accounting"
 find: "accounting"
 expect: "events [change-log login];"
</custom_item>
```

```
<custom_item>
 type: SHOW_CONFIG_CHECK
 description: "6.2.2 Require Archive Site"
```

```
 hierarchy: "system archival configuration archive-sites"
 match: "scp://"
 number_of_lines: "^([1-9]|[0-9][0-9]+)+$"
</custom_item>
```

```
<custom_item>
 type: SHOW_CONFIG_CHECK
 description: "4.7.1 Require BFD Authentication (where BFD is used)"
 hierarchy: "protocols"
 match: "authentication"
 except: "loose"
 number_of_lines: "^2$"
 check_option: CAN_BE_NULL
</custom_item>
```

```
<custom_item>
 type: SHOW_CONFIG_CHECK
 description: "4.3.1 Require MD5 Neighbor Authentication (where OSPF is used)"
 property: "ospf"
 hierarchy: "interface detail"
 match: "Auth type MD5"
 number_of_lines: "^([1-9]|[0-9][0-9]+)+$"
 check_option: CAN_BE_NULL
</custom_item>
```

Conditions

Il est possible de définir la logique **`if`**/**`then`**/**`else`** dans la stratégie de contrôle Juniper. Ceci permet à l'utilisateur final d'utiliser un seul fichier capable de traiter plusieurs configurations.

La syntaxe pour exécuter les conditions est la suivante :

```
<if>
  <condition type:"or">
      < Insert your audit here >
 </condition>
 <then>
      < Insert your audit here >
 </then>
 <else>
      < Insert your audit here >
 </else>
</if>
```

Exemple :

```
<if>
 <condition type: "OR">

<custom_item>
 type: CONFIG_CHECK
 description: "Configure Syslog Host"
```

```
 regex: "syslog host [0-9\.]+"
 not_expect: "syslog host 1.2.3.4"
</custom_item>

 </condition>
 <then>
  <report type: "PASSED">
   description: "Configure Syslog Host."
  </report>
 </then>
 <else>
<custom_item>
  type: CONFIG_CHECK
  description: "Configure Syslog Host"
  regex: "syslog host [0-9\.]+"
  not_expect: "syslog host 1.2.3.4"
</custom_item>

 </else>
</if>
```

La condition n'est jamais indiquée dans le rapport, qu'elle ait échoué ou non, car il s'agit d'une vérification « muette ».

Les conditions peuvent être du type « **and** » ou « **or** ».

Rapports

Peut être exécuté dans une instruction <then> ou <else> pour parvenir à la condition PASSED/FAILED voulue.

```
<if>
 <condition type: "OR">
 <custom_item>
  type: CONFIG_CHECK
  description: "Configure Syslog Host"
  regex: "syslog host [0-9\.]+"
  not_expect: "syslog host 1.2.3.4"
</custom_item>
 </condition>
 <then>
  <report type: "PASSED">
   description: "Configure Syslog host"
  </report>
 </then>
 <else>
 <report type: "FAILED">
   description: "Configure Syslog host"
  </report>
 </else>
</if>
```

PASSED (Réussite), WARNING (Avertissement) et FAILED (Échec) sont des valeurs acceptables pour « report type ».

Référence des fichiers de conformité d'audit pour la configuration Check Point GAiA

Cette section décrit le format et les fonctions de contrôles de Check Point GAiA (Check Point GAiA) ainsi que les raisons de chaque paramètre.

Emploi des guillemets :

Les guillemets simples et doubles sont interchangeables autour des champs d'audit, sauf dans les deux cas suivants :

1. Dans les contrôles de conformité Windows où des champs spéciaux tels que CRLF, doivent être interprétés littéralement, utilisez des guillemets simples. Tout champ intégré qui doit être interprété comme une chaîne doit être inclus dans une séquence d'échappement.

Par exemple :
```
expect: 'First line\r\nSecond line\r\nJohn\'s Line'
```

2. Des guillemets doubles sont requis lorsque vous utilisez les fichiers Windows « include_paths » et « exclude_paths ».

Si des chaînes sont utilisées dans tout type de champ (description, value_data, regex, etc.) contenant des guillemets simples ou doubles, il y a deux façons de les traiter :

a. Utilisez le type de guillemets opposés pour les guillemets extérieurs.

Par exemple :
```
expect: "This is John's Line"
expect: 'We are looking for a double-quote-".*'
```

b. Échappez tout guillemet intégré avec une barre oblique inverse (guillemets doubles uniquement).

Par exemple :
```
expect: "\"Text to be searched\""
```

Check Type: CONFIG_CHECK

Les contrôles de conformité Check Point sont placés entre crochets dans l'encapsulation **`custom_item`** et CONFIG_CHECK. Ils sont traités comme les autres fichiers **`.audit`** et ils s'appliquent aux systèmes qui utilisent le système d'exploitation Check Point GAiA. La vérification « CONFIG_CHECK » comprend au moins deux mots clés. Les mots clés **`type`** et **`description`** sont obligatoires et ils sont suivis d'un ou de plusieurs mots clés. Le contrôle vérifie la sortie de la commande « **`show config`** », qui est par défaut au format « set ».

Mots clés

Le tableau suivant indique comment chaque mot clé peut être utilisé dans les contrôles de conformité GAiA :

Mot clé	Exemple d'utilisation et paramètres pris en charge
`type`	« CHECK_CONFIG » détermine si l'élément de config spécifié existe dans la sortie « show configuration » de GAiA.
`description`	Le mot clé « **`description`** » permet d'ajouter une brève description du contrôle en cours. Il est fortement recommandé que le champ **`description`** soit unique et qu'aucun autre contrôle ne possède le même champ **`description`**. SecurityCenter de Tenable utilise ce champ pour produire automatiquement un numéro d'ID de plugin

	unique basé sur le champ **description**. Exemple : `description: "1.0 Require strong Password Controls - 'min-password-length >= 8'"`
info	Le mot clé « **info** » est utilisé pour ajouter une description plus détaillée au contrôle en cours. La raison de ce contrôle pourrait notamment être une réglementation, une URL avec plus d'informations ou une stratégie d'entreprise. Plusieurs champs **info** peuvent être ajoutés sur des lignes séparées pour formater le texte sous forme de paragraphe. Il n'existe pas de limite prédéfinie pour le nombre de champs **info** qui peuvent être utilisés.  Chaque balise « **info** » doit être écrite sur une ligne séparée sans saut de ligne. Si plusieurs lignes sont requises (par exemple, pour des raisons de formatage), ajoutez des balises « **info** » supplémentaires. Exemple : `info: "Enable palindrome-check on passwords"`
severity	Le mot clé « **severity** » spécifie la gravité du contrôle en cours. Exemple : `severity: MEDIUM` La gravité peut être paramétrée sur HIGH (Élevée), MEDIUM (Moyenne) ou LOW (Faible).
regex	Le mot clé « **regex** » permet de chercher le paramètre d'élément de configuration pour trouver une expression rationnelle particulière. Exemple : `regex: "set snmp .+"` Les caractères génériques suivants nécessitent un traitement spécial : + \ * ( ) ^ Échapper deux fois ces caractères avec deux barres obliques inverses « \\ » ou les placer entre crochets « [] » s'ils doivent être interprétés littéralement. D'autres caractères tels que ceux indiqués ci-dessous nécessitent seulement une barre oblique inverse pour être interprétés littéralement : . ? " ' La raison provient de la façon dont le compilateur traite ces caractères. Si une balise « regex » est définie pour un contrôle, mais pas une balise « expect », « not_expect » ou « number_of_lines », le contrôle signale simplement toutes les lignes correspondant au mot clé regex.
expect	Ce mot clé permet de vérifier l'élément de configuration qui correspond à la balise « **regex** ». Si la balise « **regex** » n'est pas utilisée, il recherche la chaîne « **expect** » dans l'ensemble de config. Le contrôle renvoie un résultat concluant lorsque la ligne de config trouvée par « **regex** » correspond à la balise « **expect** » ou, si « **regex** » n'est pas défini, lorsque la chaîne « **expect** » est trouvée dans config.

	Exemple :
	`regex: "set password-controls complexity"`
	`expect: "set password-controls complexity [1-4]"`
	
	Dans le cas ci-dessus, la balise « **expect** » garantit que la complexité est définie avec une valeur comprise entre 1 et 4.

not_expect	Ce mot clé permet de rechercher les éléments de configuration qui ne devraient pas figurer dans la configuration.
	
	Il a l'action contraire de « **expect** ». Le contrôle est concluant lorsque la ligne de config trouvée par « **regex** » ne correspond pas à la balise « **not_expect** » ou, si « **regex** » n'est pas défini, lorsque la chaîne « **not_expect** » n'est pas trouvée dans config.
	
	Exemple :
	`regex: "set password-controls password-expiration"`
	`not_expect: "set password-controls password-expiration never"`
	
	Dans le cas ci-dessus, la balise « **not_expect** » garantit que password-controls n'est pas paramétré sur « never ».

Exemples de CONFIG_CHECK

Les exemples qui suivent illustrent l'utilisation de CONFIG_CHECK pour un périphérique Check Point :

```
<custom item>
 type: CONFIG_CHECK
 description: "1.0 Require strong Password Controls - 'min-password-length >= 8'"
 regex: "set password-controls min-password-length"
 expect: "set password-controls min-password-length ([8-9]|[0-9][0-9]+)"
 info: "Require Password Lengths greater than or equal to 8."
</custom_item>
```

```
<custom_item>
 type: CONFIG_CHECK
 description: "1.0 Require strong Password Controls - 'password-expiration != never'"
 regex: "set password-controls password-expiration"
 not_expect: "set password-controls password-expiration never"
 info: "Allow passwords to expire"
</custom_item>
```

```
<custom_item>
 type: CONFIG_CHECK
 description: "2.13 Secure SNMP"
 regex: "set snmp .+"
 severity: MEDIUM
 info: "Manually review SNMP settings."
</custom_item>
```

Conditions

Il est possible de définir la logique **if/then/else** dans la stratégie de contrôle Check Point. Ceci permet à l'utilisateur final d'utiliser un seul fichier capable de traiter plusieurs configurations.

La syntaxe pour exécuter les conditions est la suivante :

```
<if>
  <condition type:"or">
      < Insert your audit here >
 </condition>
 <then>
      < Insert your audit here >
 </then>
 <else>
      < Insert your audit here >
 </else>
</if>
```

Exemple :

```
<if>
 <condition type: "OR">
 <custom_item>
  type: CONFIG_CHECK
  description: "2.6 Install and configure Encrypted Connections to devices - 'telnet'"
  regex: "set net-access telnet"
  expect: "set net-access telnet off"
  info: "Do not use plain-text protocols."
</custom_item>
 </condition>
 <then>
  <report type: "PASSED">
   description: "Telnet is disabled"
  </report>
 </then>
 <else>
  <custom_item>
  type: CONFIG_CHECK
  description: "2.6 Install and configure Encrypted Connections to devices - 'telnet'"
  regex: "set net-access telnet"
  expect: "set net-access telnet off"
  info: "Do not use plain-text protocols."
</custom_item>
 </else>
</if>
```

La condition n'est jamais indiquée dans le rapport, qu'elle ait échoué ou non, car il s'agit d'une vérification « muette ».

Les conditions peuvent être du type « **and** » ou « **or** ».

Rapports

Peut être exécuté dans une instruction <then> ou <else> pour parvenir à la condition PASSED/FAILED (Réussite/Échec) voulue.

```
<if>
 <condition type: "OR">
 <custom_item>
  type: CONFIG_CHECK
```

```
  description: "2.6 Install and configure Encrypted Connections to devices - 'telnet'"
  regex: "set net-access telnet"
  expect: "set net-access telnet off"
  info: "Do not use plain-text protocols."
</custom_item>
 </condition>
 <then>
  <report type: "PASSED">
   description: "Telnet is disabled"
  </report>
 </then>
 <else>
 <report type: "FAILED">
   description: "Telnet is disabled"
  </report>
 </else>
</if>
```

PASSED (Réussite), WARNING (Avertissement) et FAILED (Échec) sont des valeurs acceptables pour « report type ».

Référence des fichiers de conformité d'audit pour la configuration du pare-feu Palo Alto

Les contrôles de conformité pour Palo Alto sont différents des autres contrôles de conformité. L'une des principales différences de ces contrôles réside dans l'utilisation intensive des transformations XLS (`XSL Transforms` - XSLT) pour extraire les éléments d'information pertinents (voir l'Annexe C pour plus d'informations). Les réponses du pare-feu Palo Alto sont au format XML pour la plupart des demandes API, si bien que XSLT s'impose comme la méthode la plus efficace pour les audits. Si vous ne maîtrisez pas XSLT, vous pouvez le voir comme une méthode qui permet d'interroger un fichier XML afin d'en extraire les données recherchées, au format voulu. En termes simples, XSLT remplit le même rôle que SQL pour les bases de données.

L'audit Palo Alto prend en charge deux types de contrôles : AUDIT_XML et AUDIT_REPORTS.

AUDIT_XML

Voici un exemple de contrôle AUDIT_XML Palo Alto :

```
<custom_item>
 type: AUDIT XML
 description: "Palo Alto Security Settings - 'fips-mode = on'"
 info: "Fips-mode should be enabled."
 api_request_type: "op"
 request: "<show><fips-mode></fips-mode></show>"
 xsl_stmt: "<xsl:template match=\"/\">"
 xsl_stmt: " <xsl:apply-templates select=\"//result\"/>"
 xsl_stmt: "</xsl:template>"
 xsl_stmt: "<xsl:template match=\"//result\">"
 xsl_stmt: "fips-mode: <xsl:value-of select=\"text()\"/>"
 regex: "fips-mode:[\\s\\t]+"
 expect : "fips-mode:[\\s\\t]+on"
</custom_item>
```

Cet audit se compose de quatre parties principales :

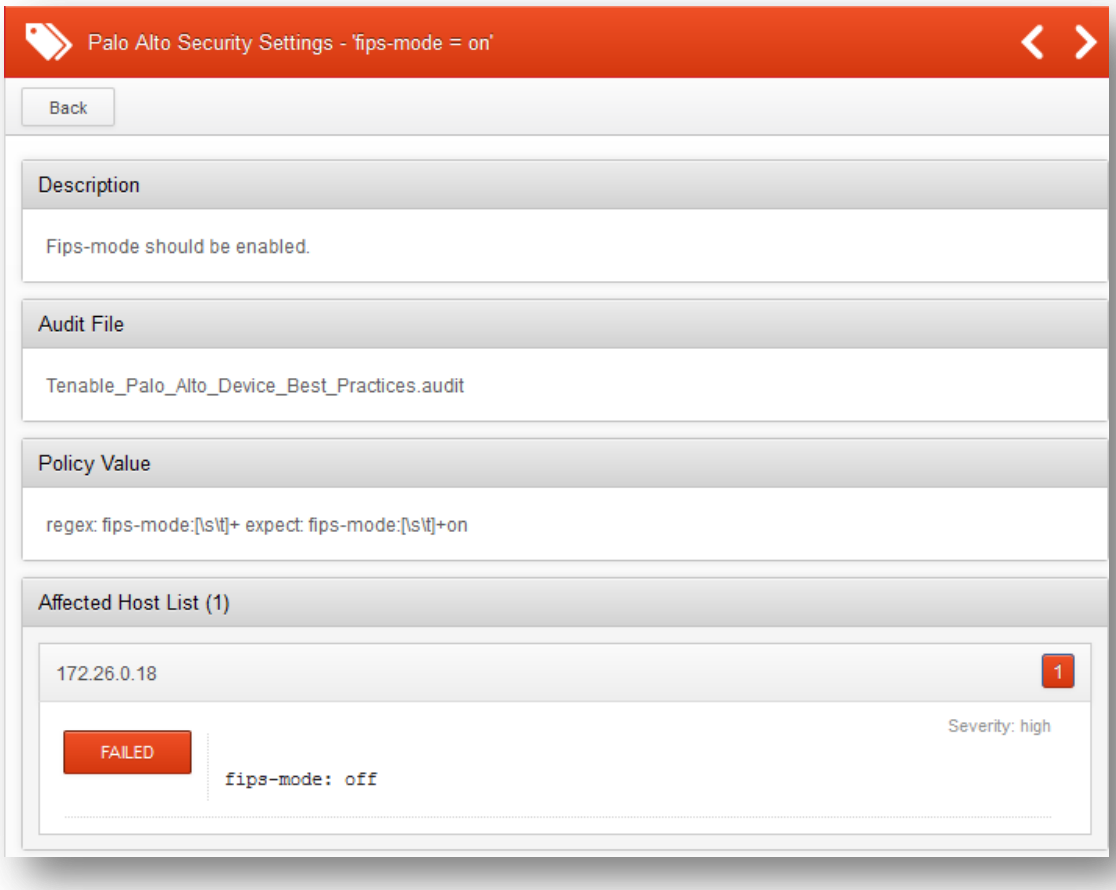
1. Le mot clé **`type`** indique le type d'audit (dans ce cas, XML) et une **`description`** de l'audit. Le mot clé **`info`** keyword permet d'inclure le texte pertinent dans le rapport.

2. Le mot clé **`api_request_type`** indique le type de demande (op == config opérationnelle), et la demande est la demande réelle que nous allons exécuter. Il s'agit actuellement du seul type de demande pris en charge.

3. Le mot clé **`xsl_stmt`** permet de définir la transformation XSL que nous allons appliquer au XML retourné après l'exécution de la demande API.

4. Enfin, les mots clés **`regex`** et **`expect`** permettent d'effectuer l'audit de conformité/configuration.

L'exemple de contrôle ci-dessus génère le rapport suivant dans Nessus :



AUDIT_REPORTS

L'une des fonctions particulièrement utiles d'un pare-feu Palo Alto est qu'il surveille en permanence le profil de son réseau, générant plus de 40 rapports prédéfinis sur une base quotidienne. Des rapports tels que Top Applications, Top Attackers et Spyware Infected Hosts. Les administrateurs peuvent également générer des rapports dynamiques à leur convenance (par exemple, pour l'heure qui vient de s'écouler). Nessus peut désormais interroger directement ces rapports et les inclure dans un rapport Nessus.

Cette fonction présente deux avantages. D'abord, les utilisateurs ne sont pas tenus de naviguer dans des interfaces différentes pour obtenir les mêmes données. Ensuite, elle permet d'effectuer l'audit du rapport. Par exemple, si vous ne voulez pas que l'application Facebook soit utilisée dans le réseau, les administrateurs peuvent générer un échec de rapport si Facebook apparaît dans le rapport Top Applications. Par exemple :

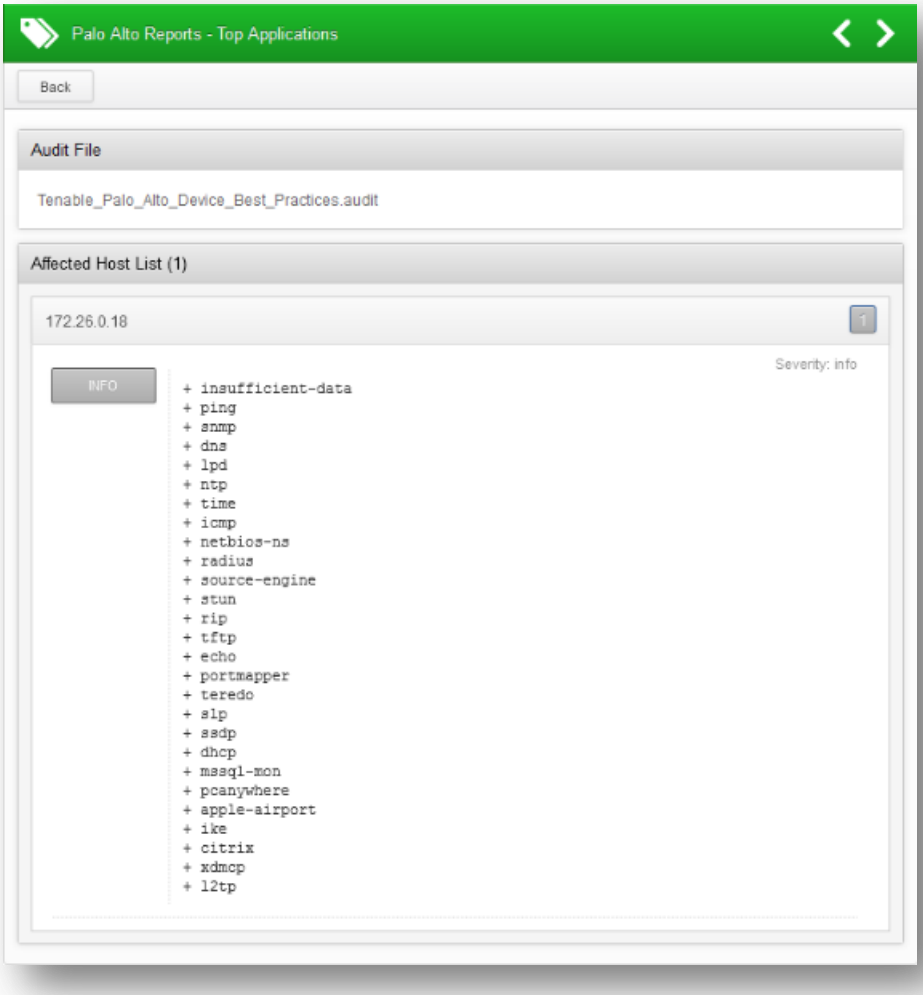
```
<custom_item>
 type: AUDIT_REPORTS
 description: "Palo Alto Reports - Top Applications"
```

```
 request: "&reporttype=predefined&reportname=top-applications"
 xsl_stmt: "<xsl:template match=\"result\">"
 xsl_stmt: "<xsl:for-each select=\"entry\">"
 xsl_stmt: "+ <xsl:value-of select=\"name\"/>"
 xsl_stmt: "</xsl:for-each>"
 check_option: CAN_BE_NULL
</custom_item>
```

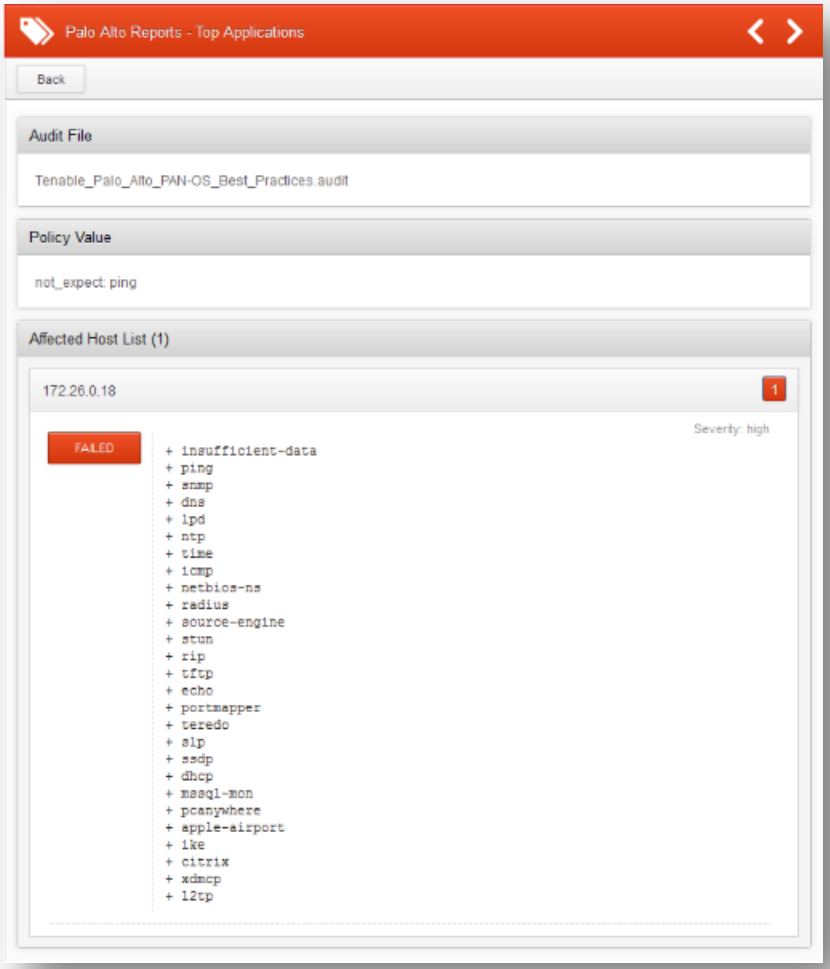
Ce rapport peut être modifié afin d'utiliser un mot clé **`not_expect`** :

```
<custom_item>
 type: AUDIT_REPORTS
 description: "Palo Alto Reports - Top Applications"
 request: "&reporttype=predefined&reportname=top-applications"
 xsl_stmt: "<xsl:template match=\"result\">"
 xsl_stmt: "<xsl:for-each select=\"entry\">"
 xsl_stmt: "+ <xsl:value-of select=\"name\"/>"
 xsl_stmt: "</xsl:for-each>"
 not_expect: "ping"
 check_option: CAN_BE_NULL
</custom_item>
```

Le premier exemple renvoie un rapport similaire à celui-ci :


Palo Alto Reports - Top Applications
Back
Audit File
Tenable_Palo_Alto_Device_Best_Practices.audit
Affected Host List (1)
172.26.0.18
1
Severity: info
INFO
+ insufficient-data
+ ping
+ snmp
+ dns
+ lpd
+ ntp
+ time
+ icmp
+ netbios-ns
+ radius
+ source-engine
+ stun
+ rip
+ tftp
+ echo
+ portmapper
+ teredo
+ slp
+ ssdp
+ dhcp
+ mssql-mon
+ pcanywhere
+ apple-airport
+ ike
+ citrix
+ xdmcp
+ l2tp

Le deuxième exemple renvoie un rapport qui échoue :



Mots clés

Les audits Palo Alto prennent en charge les mots clés suivants :

Mot clé	Description
type	Ce mot clé doit toujours être paramétré sur AUDIT_XML ou AUDIT_REPORTS.
description	Information utilisée comme titre pour les vulnérabilités uniques de conformité dans SecurityCenter. C'est aussi le premier groupe de données signalé par Nessus.
info	Ce mot clé est utilisé pour ajouter une description plus détaillée au contrôle en cours. Plusieurs champs **info** sont autorisés sans limite prédéfinie. Le contenu du champ **info** doit toujours être inclus entre guillemets.
api_request_type	Ce mot clé décrit le type de demande. L'API Palo Alto prend en charge six types de demandes : keygen, op, commit, reports, export et config. Pour ce plugin, seul le type de demande **op** est exposé.

request	Ce mot clé spécifie la demande à exécuter sur le pare-feu. Le résultat de chaque demande est mis en cache pour éviter que les demandes suivantes n'entraînent une autre demande. De plus, l'audit Tenable par défaut ne comprend que 9 contrôles dans le cas d'un contrôle AUDIT_REPORTS. Pour inclure davantage de rapports, les utilisateurs sont encouragés à créer de nouveaux contrôles et à remplacer le mot clé request par l'URL de l'API REST après `type=report`. Par exemple : `/api/?type=report&reporttype=predefined&reportname=hruser-top-url-categories`
regex	Ce mot clé permet de chercher les éléments qui correspondent à une expression regex particulière. Si un mot clé **regex** est défini pour un contrôle, mais pas un mot clé **expect** ou **not_expect**, le contrôle signale simplement toutes les lignes correspondant au mot clé regex.

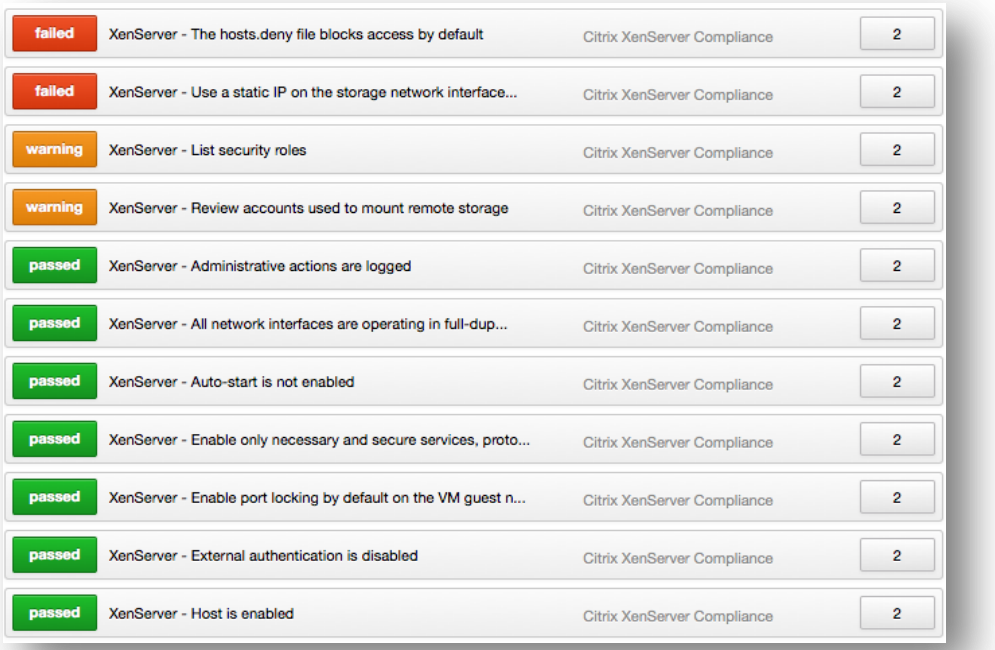
La conformité d'un contrôle peut être déterminée en comparant la sortie du contrôle au mot clé **expect** ou **not_expect**. Il ne peut y avoir qu'une seule balise de test de conformité (par exemple, **expect** ou **not_expect**, mais pas **expect** et **not_expect**).

Mot clé	Description
expect	Ce mot clé permet de vérifier l'élément de config qui correspond au mot clé **regex**. Si le mot clé **regex** n'est pas utilisé, il recherche la chaîne **expect** dans l'ensemble de config. Le contrôle renvoie un résultat concluant lorsque la ligne de config trouvée par **regex** correspond à la chaîne **expect** ou, si **regex** n'est pas défini, lorsque la chaîne **expect** est trouvée dans config.
not_expect	Ce mot clé permet de rechercher les éléments de configuration qui **ne doivent pas** figurer dans la configuration. Il a l'action contraire de **expect**. Le contrôle est concluant tant que la ligne de config trouvée par **regex** ne correspond pas à la chaîne **not_expect** ou, si le mot clé **regex** n'est pas défini, tant que la chaîne **not_expect** n'est pas trouvée dans config.

Référence des fichiers de conformité d'audit pour la configuration Citrix XenServer

Les contrôles de conformité pour Citrix XenServer reposent dans une large mesure sur la section Référence des fichiers de conformité d'audit pour la configuration UNIX, avec une exception. Un audit supplémentaire intitulé AUDIT_XE permet d'effectuer l'audit des correctifs. Les types de contrôles suivants sont disponibles pour les audits XenServer :

- FILE_CHECK_NOT
- PROCESS_CHECK
- FILE_CONTENT_CHECK
- FILE_CONTENT_CHECK_NOT
- CMD_EXEC
- GRAMMAR_CHECK
- RPM_CHECK
- CHKCONFIG
- XINETD_SVC
- AUDIT_XE



Check Type: AUDIT_XE

Voici un exemple de contrôle XenServer AUDIT_XE check:

```
<custom_item>
 type: AUDIT_XE
 description: "List halted VMs"
 info: "Current guest VM status."
 reference: "PCI|2.2.3,SANS-CSC|1"
 cmd: "/usr/bin/xe vm-list power-state=halted params=uuid,name-label,power-state"
 # You can ignore VMs expected to be halted by entering their UUID here
 # Example ignore
 # ignore: "669e1681-2968-7435-c88e-663501f7d8f3"
</custom_item>
```

Mots clés

Le tableau suivant indique comment chaque mot clé peut être utilisé dans les contrôles de conformité de Citrix XenServer :

Mot clé	Exemple d'utilisation et paramètres pris en charge
type	AUDIT_XE
description	Ce mot clé fournit une brève description du contrôle en cours. Il est nécessaire que le champ description soit unique et deux contrôles quelconques ne doivent pas avoir le même champ de description. En effet, SecurityCenter de Tenable utilise ce champ pour produire automatiquement un numéro d'ID de plugin unique basé sur le champ de description. Exemple :

	`description: "List running VMs"`
info	Ce mot clé est utilisé pour ajouter une description plus détaillée au contrôle en cours. Plusieurs champs info sont autorisés sans limite prédéfinie. Le contenu du champ info doit toujours être inclus entre guillemets. Exemple : `info: "The allocated virtual CPUs (VCPU) should be reviewed. Desired settings depend on workload and operating system type."`
see_also	Ce mot clé permet d'inclure des liens qui peuvent fournir des informations utiles sur un contrôle. Exemple : `see_also: "http://support.citrix.com/article/CTX137828"`
reference	Ce mot clé permet d'inclure des références croisées pour les contrôles d'audit. Exemple : `reference: "PCI|2.2.3,SANS-CSC|1"`
solution	Ce mot clé fournit le texte à inclure dans le texte de solution pour remédier à un problème de conformité.
severity	Ce mot clé permet aux utilisateurs de définir la gravité du contrôle. La gravité peut être paramétrée sur HIGH (Élevée), MEDIUM (Moyenne) ou LOW (Faible). Exemple : `severity: MEDIUM`
cmd	Ce mot clé spécifie la commande **xe** en cours d'exécution sur la cible. Exemple : `cmd: "/usr/bin/xe subject-list params=all"`
regex	Ce mot clé permet d'énumérer les éléments qui correspondent à une expression regex particulière. Si un mot clé « **regex** » est défini pour un contrôle, mais pas un mot clé « **expect** » ou « **not_expect** », le contrôle signale simplement tous les éléments correspondant au mot clé regex. Exemple : `regex: "power-state.+"`

expect	Si le mot clé expect est spécifié, le contrôle réussit uniquement si tous les résultats correspondent au mot clé « **expect** »d. Si un résultat ne correspond pas au mot clé **expect**, le contrôle échouera avec aucun résultat ne correspondant au mot clé **expect**.
	
	Exemple :
	
	`<custom_item>`
	`type: AUDIT_XE`
	`description: "List Running VMs - Any non running vms."`
	`cmd: "/usr/bin/xe vm-list params=uuid,name-label,is-a-template,power-state,allowed-operations"`
	`regex: "power-state .+"`

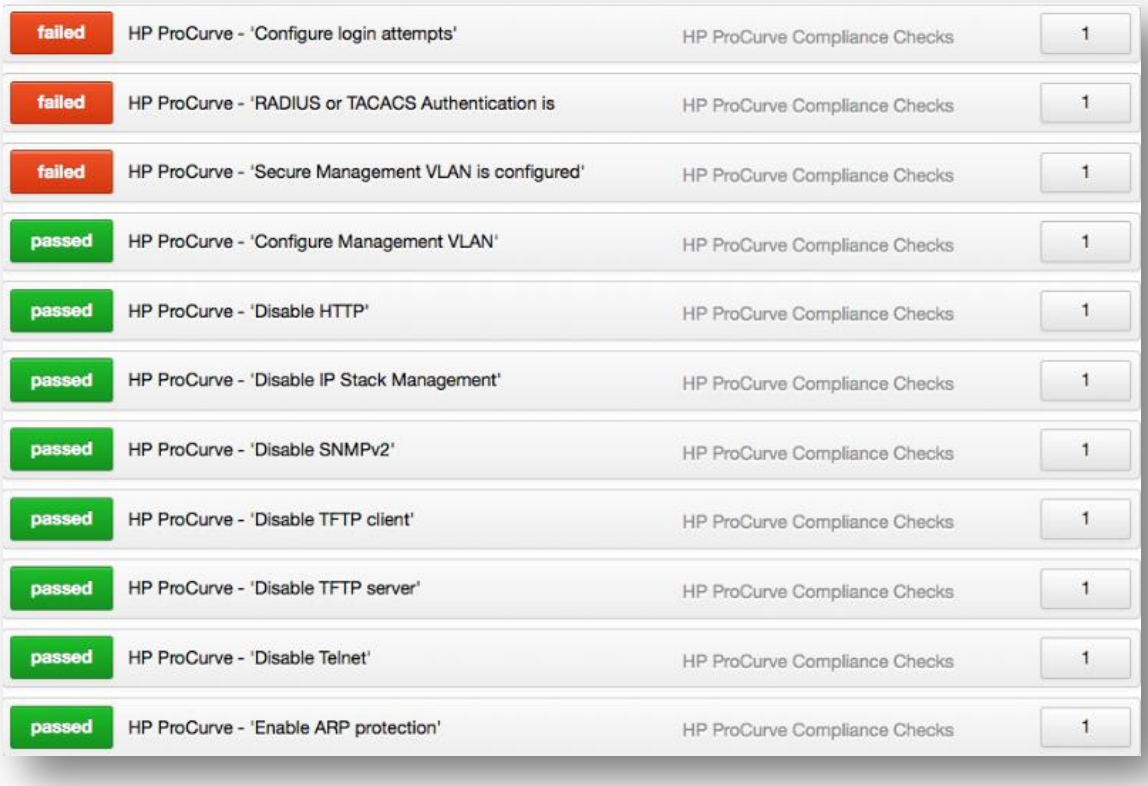
	<pre>expect: "running"
	</custom_item></pre>

not_expect	Si le mot clé not_expect n'est pas défini, le contrôle réussit à condition qu'aucun des résultats ne correspondent au mot clé regex de not_expect.
	
	Exemple :
	
	<pre><custom_item>
	type: AUDIT_XE
	description: "List Running VMs"
	cmd: "/usr/bin/xe vm-list params=uuid,name-label,is-a-
	template,power-state,allowed-operations"
	regex: "power-state .+"
	not_expect: "halted"
	</custom_item></pre>

ignore	Ce mot clé permet d'ignorer/sauter certains éléments dans le résultat.
	
	Exemple :
	
	<pre><custom_item>
	type: AUDIT_XE
	description: "List halted VMs"
	info: "Current guest VM status."
	cmd: "/usr/bin/xe vm-list power-state=halted
	params=uuid,name-label,power-state"
	# You can ignore VMs expected to be halted by entering their
	UUID here
	# Example ignore
	ignore: "669e1681-2968-7435-c88e-663501f7d8f3"
	</custom_item></pre>

Référence des fichiers de conformité d'audit pour la configuration HP ProCurve

L'audit HP ProCurve est, à bien des égards, une extension du plugin de conformité Cisco. Le fichier d'audit Tenable HP ProCurve repose sur un livre blanc HP relatif au renforcement des commutateurs ProCurve. L'audit inclut des contrôles qui permettent de déactiver des services non sécurisés et d'activer le contrôle d'accès (par exemple, TACACS, RADIUS). Les identifiants SSH valides pour un utilisateur root ou un administrateur avec privilèges complets sont requis.



Types de contrôle

Les contrôles de conformité HP ProCurve utilisent l'un des trois types de contrôles suivants. L'audit utilise la syntaxe générale suivante :

```
<custom_item>
 type: CONFIG_CHECK
 description: "Verify login authentication"
 info: "Verifies login authentication configuration"
 reference: "PCI|2.2.3,SANS-CSC|1"
 context: "line .*"
 item: "login authentication"
</custom_item>
```

Mots clés

Le tableau suivant indique comment chaque mot clé peut être utilisé dans les contrôles de conformité de HP ProCurve :

Mot clé	Exemple d'utilisation et paramètres pris en charge
type	CONFIG_CHECK CONFIG_CHECK_NOT RANDOMNESS_CHECK
description	Ce mot clé fournit une brève description du contrôle en cours. Il est nécessaire que le champ description soit unique et deux contrôles quelconques ne doivent pas avoir le même champ de description. En effet, SecurityCenter de Tenable utilise ce champ

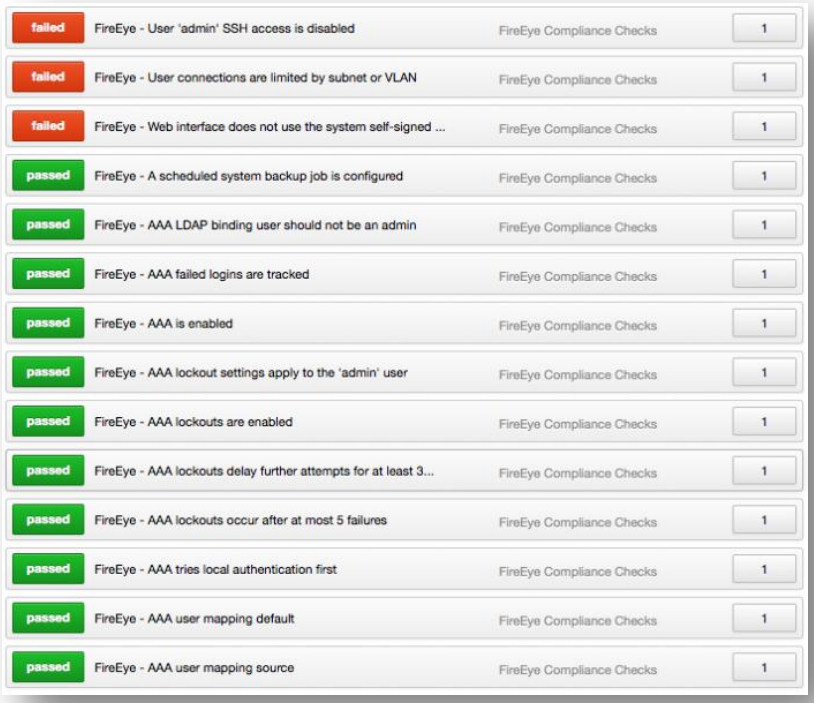
	pour produire automatiquement un numéro d'ID de plugin unique basé sur le champ de description. Exemple : `description: " Verify login authentication"`
info	Ce mot clé est utilisé pour ajouter une description plus détaillée au contrôle en cours. Plusieurs champs info sont autorisés sans limite prédéfinie. Le contenu du champ info doit toujours être inclus entre guillemets. Exemple : `info: "Verifies login authentication configuration."`
see_also	Ce mot clé permet d'inclure des liens qui peuvent fournir des informations utiles sur un contrôle. Exemple : `see_also: "http://www.hp.com/rnd/support/faqs/1800.htm"`
reference	Ce mot clé permet d'inclure des références croisées pour les contrôles d'audit. Exemple : `reference: "PCI|2.2.3,SANS-CSC|1"`
solution	Ce mot clé fournit le texte à inclure dans le texte de solution pour remédier à un problème de conformité. Exemple : `solution: "Modify the configuration to add missing line"`
severity	Ce mot clé permet aux utilisateurs de définir la gravité du contrôle. La gravité peut être paramétrée sur HIGH (Élevée), MEDIUM (Moyenne) ou LOW (Faible). Exemple : `severity: MEDIUM`
regex	Ce mot clé permet d'énumérer les éléments qui correspondent à une expression regex particulière. Si un mot clé « **regex** » est défini pour un contrôle, mais pas un mot clé « **expect** » ou « **not_expect** », le contrôle signale simplement tous les éléments correspondant au mot clé regex. Exemple : `regex: "power-state.+"`
item	Ce mot clé permet d'effectuer une recherche dans les lignes trouvées par regex. Si aucun regex n'a été fourni, toutes les lignes seront contrôlées. Exemple : `regex: "power"`
context	Ce mot clé permet d'effectuer une recherche dans un contexte spécifique. Un contexte est défini par une ligne justifiée à gauche, suivie de toutes les lignes préfixées par un espace blanc. Exemple : `context: "line .*"` Voici un exemple d'élément de config, qui pourrait faire l'objet d'un audit basé sur

	l'exploitation du contexte :
	
	`vlan 1`
	`  name "DEFAULT_VLAN"`
	`  untagged 2-24`
	`  ip address dhcp-bootp`
	`  no untagged 1`
	`  exit`
	
	`<item>`
	` type: CONFIG_CHECK`
	` description: "HP ProCurve - 'dhcp-bootp'"`
	` context: "vlan 1"`
	` item: "ip address dhcp-bootp"`
	`</item>`
	
	Le contrôle ci-dessous garantit que « `ip address dhcp-bootp` » est défini pour le contexte « `vlan 1` ».

`min_occurrences`	Ce mot clé spécifie le nombre minimum d'occurrences du contrôle. Exemple : `min_occurrences: 3`
`max_occurrences`	Similaire à **`min_occurrences`**, mais le nombre minimum est remplacé par un nombre maximum.
`required`	Ce mot clé permet de spécifier si une correspondance de contrôle est requise ou non. Le champ required peut avoir la valeur YES, NO, ENABLED ou DISABLED. Exemple : `required: YES`

Référence des fichiers de conformité d'audit pour la configuration FireEye

L'audit FireEye repose sur la documentation produit de FireEye et sur les consignes de critères courantes. L'audit inclut des contrôles pour les audits, l'identification et l'authentification, la gestion des périphériques, l'interface de gestion intelligente de matériel (IPMI), les services activés, le cryptage et la configuration des systèmes de détection des logiciels malveillants. Les identifiants SSH valides pour un utilisateur root ou un administrateur avec privilèges complets sont requis.



Types de contrôle

Les contrôles de conformité FireEye utilisent l'un des trois types de contrôles suivants. L'audit utilise la syntaxe générale suivante :

```
<item>
 type: CONFIG_CHECK
 description: "Specific user privs"
 info: "Expect to fail on running config since not all username lines match"
 regex: "username .+"
 expect: " username egossell capability admin"
</item>
```

Mots clés

Le tableau suivant indique comment chaque mot clé peut être utilisé dans les contrôles de conformité de HP ProCurve :

Mot clé	Exemple d'utilisation et paramètres pris en charge
type	CONFIG_CHECK CONFIG_CHECK_NOT RANDOMNESS_CHECK
description	Ce mot clé fournit une brève description du contrôle en cours. Il est nécessaire que le champ description soit unique et deux contrôles quelconques ne doivent pas avoir le même champ de description. En effet, SecurityCenter de Tenable utilise ce champ pour produire automatiquement un numéro d'ID de plugin unique basé sur le champ de description.

	Exemple : `description: " Verify login authentication"`
info	Ce mot clé est utilisé pour ajouter une description plus détaillée au contrôle en cours. Plusieurs champs info sont autorisés sans limite prédéfinie. Le contenu du champ info doit toujours être inclus entre guillemets. Exemple : `info: "Verifies login authentication configuration."`
see_also	Ce mot clé permet d'inclure des liens qui peuvent fournir des informations utiles sur un contrôle. Exemple : `see_also: "http://www.fireeye.com/support/"`
reference	Ce mot clé permet d'inclure des références croisées pour les contrôles d'audit. Exemple : `reference: "PCI|2.2.3,SANS-CSC|1"`
solution	Ce mot clé fournit le texte à inclure dans le texte de solution pour remédier à un problème de conformité. Exemple : `solution: "Modify the configuration to add missing line"`
severity	Ce mot clé permet aux utilisateurs de définir la gravité du contrôle. La gravité peut être paramétrée sur HIGH (Élevée), MEDIUM (Moyenne) ou LOW (Faible). Exemple : `severity: MEDIUM`
regex	Ce mot clé permet d'énumérer les éléments qui correspondent à une expression regex particulière. Si un mot clé « **regex** » est défini pour un contrôle, mais pas un mot clé « **expect** » ou « **not_expect** », le contrôle signale simplement tous les éléments correspondant au mot clé regex. Exemple : `regex: "power-state.+"`
expect	Ce mot clé permet d'effectuer une recherche dans les lignes trouvées par regex. Pour que le contrôle réussisse, toutes les lignes trouvées par regex doivent correspondre au paramètre expect. Si aucun regex n'a été fourni, toutes les lignes seront contrôlées mais il suffit d'en trouver une seule. Exemple : `regex: "power"`
not_expect	Similaire à **expect**, mais le contrôle échoue si une correspondance quelconque est trouvée. Si **expect** et **not_expect** sont omis, toutes les lignes applicables sont signalées en tant que message d'information.
min_occurrences	Ce mot clé spécifie le nombre minimum d'occurrences du contrôle. Exemple : `min_occurrences: 3`

max_occurrences	Similaire à **min_occurrences**, mais le nombre minimum est remplacé par un nombre maximum.
required	Ce mot clé permet de spécifier si une correspondance de contrôle est requise ou non. Le champ required peut avoir la valeur YES, NO, ENABLED ou DISABLED. Exemple : `required: YES`

cmd	Ce mot clé permet d'exécuter une commande show.
	
	Exemple :
	`cmd: "show version"`
	
	Seules les commandes « **show** » sont autorisées.
	
	`<item>`
	` type: CONFIG_CHECK`
	` cmd: "show version"`
	` description: "Show Product version"`
	` regex: "Product model:"`
	` expect: "1234"`
	`</item>`

Référence des fichiers de conformité d'audit pour la configuration de base de données

Cette section décrit le format et les fonctions de contrôles de conformité de base de données, ainsi que les raisons de chaque paramètre.

Emploi des guillemets :

Les guillemets simples et doubles sont interchangeables autour des champs d'audit, sauf dans les deux cas suivants :

1. Dans les contrôles de conformité Windows où des champs spéciaux tels que CRLF, doivent être interprétés littéralement, utilisez des guillemets simples. Tout champ intégré qui doit être interprété comme une chaîne doit être inclus dans une séquence d'échappement.

Par exemple :
`expect: 'First line\r\nSecond line\r\nJohn\'s Line'`

2. Des guillemets doubles sont requis lorsque vous utilisez les fichiers Windows « include_paths » et « exclude_paths ».

Si des chaînes sont utilisées dans tout type de champ (description, value_data, regex, etc.) contenant des guillemets simples ou doubles, il y a deux façons de les traiter :

a. Utilisez le type de guillemets opposés pour les guillemets extérieurs.

Par exemple :
```
expect: "This is John's Line"
expect: 'We are looking for a double-quote-".*'
```

b. Échappez tout guillemet intégré avec une barre oblique inverse (guillemets doubles uniquement).

Par exemple :
`expect: "\"Text to be searched\""`

Type de contrôle

Tous les contrôles de conformité de base de données doivent être mis entre crochets avec l'encapsulation **check_type** et la désignation « Database » (Base de données). Ceci est requis pour distinguer les fichiers **.audit** conçus spécifiquement pour les bases de données des autres types d'audit de conformité. Le champ **check_type** nécessite deux paramètres supplémentaires :

- **db_type**
- **version**

Les types de base de données disponibles pour les audits incluent :

- SQLServer
- Oracle
- MySQL
- PostgreSQL
- DB2
- Informix

La **version** est pour le moment toujours paramétrée sur « 1 ».

Exemple :

```
<check_type: "Database" db_type:"SQLServer" version:"1">
```

Mots clés

Le tableau suivant indique comment chaque mot clé peut être utilisé dans les contrôles de conformité de base de données :

Mot clé	Exemple d'utilisation et paramètres pris en charge
type	`SQL_POLICY`
description	Ce mot clé permet d'ajouter une brève description du contrôle en cours. Il est fortement recommandé que le champ **description** soit unique et qu'aucun autre contrôle ne possède le même champ. SecurityCenter de Tenable utilise ce champ pour produire automatiquement un numéro d'ID de plugin unique basé sur le champ **description**. Exemple : `description: "DBMS Password Complexity"`
info	Ce mot clé est utilisé pour ajouter une description plus détaillée du contrôle en cours comme une réglementation, une URL, une stratégie d'entreprise ou une autre raison pour laquelle le paramètre est requis. Plusieurs champs **info** peuvent être ajoutés sur des lignes séparées pour formater le texte sous forme de paragraphe. Il n'existe pas de limite prédéfinie pour le nombre de champs **info** qui peuvent être utilisés. Exemple :

	`info: "Checking that \"password complexity\" requirements are enforced for systems using SQL Server authentication."`

sql_request	Ce mot clé est utilisé pour déterminer la demande **SQL** réelle à soumettre à la base de données. Des tableaux de données peuvent être demandés et retournés à partir d'une demande **SQL** en utilisant des valeurs de demande/retour délimitées par la virgule.
	
	Exemple :
	`sql_request: "select name from sys.sql_logins where type = 'S' and is_policy_checked <> '1'"`
	
	Exemple :
	`sql_request: "select name, value_in_use from sys.configurations where name = 'clr enabled'"`

sql_types	Ce mot clé peut avoir deux options : **POLICY_VARCHAR** et **POLICY_INTEGER**. Utilisez **POLICY_INTEGER** pour des valeurs numériques comprises entre 0 et 2147483647 et **POLICY_VARCHAR** pour tout autre type de valeur de retour.
	
	Exemple :
	`sql_types: POLICY_VARCHAR`
	
	Exemple :
	`sql_types: POLICY_VARCHAR,POLICY_INTEGER`
	
	Pour plusieurs éléments de retour, configurez **sql_types** dans une liste séparée par une virgule pour accepter les types de données de chaque résultat de retour **SQL**. L'exemple ci-dessus indique que la première valeur de retour de l'interrogation **SQL** est **varchar** et la deuxième valeur de retour est un entier.

sql_expect	Ce mot clé est utilisé pour déterminer la valeur de retour attendue pour la demande **SQL**. Une valeur exacte incluant **NULL** ou « **0** » peut être requise. En outre, des expressions rationnelles peuvent être requises pour **POLICY_VARCHAR sql_types**.
	
	Exemple :
	`sql_expect: regex:"^.+Failure" || regex:"^.+ALL"`
	
	Exemple :
	`sql_expect: NULL`
	
	Exemple :
	`sql_expect: 0 || "0"`
	
	Les guillemets doubles sont facultatifs pour les valeurs de retour entières.
	
	Exemple :
	`sql_expect: "clr enabled",0`
	
	Un tableau de données peut être retourné à partir d'une demande **SQL** et inclus au format séparé par une virgule dans le champ **sql_expect**.

Exemples de ligne de commande

Cette section fournit quelques exemples de vérifications fréquemment utilisées pour les contrôles de conformité de base de données. Le binaire de ligne de commande **nasl** est utilisé comme moyen rapide de tester les audits sur-le-champ. Chacun des fichiers **.audit** démontrés ci-dessus peut être facilement amené dans les stratégies de scan Nessus ou

SecurityCenter. Toutefois, pour les audits rapides d'un système, les tests par ligne de commande sont plus efficaces. La commande sera exécutée chaque fois à partir du répertoire **/opt/nessus/bin** comme suit :

```
# ./nasl -t <IP> /opt/nessus/lib/nessus/plugins/database_compliance_check.nbin
```

<IP> est l'adresse IP du système à vérifier.

Selon le type de base de données vérifié, vous pouvez être invité à saisir d'autres paramètres, en plus du fichier d'audit à utiliser. Par exemple, les audits Oracle produiront une invite pour la DIS de base de données et le type de connexion Oracle :

```
Which file contains your security policy : oracle.audit
login : admin
Password :
Database type: ORACLE(0), SQL Server(1), MySQL(2), DB2(3), Informix/DRDA(4),
       PostgreSQL(5)
type : 0
sid: oracle
Oracle login type: NORMAL (0), SYSOPER (1), SYSDBA (2)
type: 2
```

Consultez votre administrateur de base de données pour les paramètres corrects de connexion de base de données.

Exemple 1 : Rechercher les connexions sans date d'expiration

Le fichier **.audit** simple ci-dessous recherche toute connexion SQL Server sans date d'expiration. S'il en trouve une, l'audit affiche un message d'erreur ainsi que la ou les connexions illégales.

```
<check_type: "Database" db_type:"SQLServer" version:"1">
<group_policy: "Login expiration check">
<custom_item>
 type: SQL_POLICY
 description: "Login expiration check"
 info: "Database logins with no expiration date pose a security threat. "
 sql_request: "select name from sys.sql_logins where type = 'S' and
       is_expiration_checked = 0"
 sql_types: POLICY_VARCHAR
 sql_expect: NULL
</custom_item>
</group_policy
</check_type>
```

Lors de l'exécution de cette commande, la sortie suivante est attendue pour un système conforme :

```
"Login expiration check": [PASSED]
```

Les exigences de conformité spécifient en général que les connexions de base de données doivent avoir une date d'expiration.

Un échec de l'audit renverrait la sortie suivante :

```
"Login expiration check": [FAILED]
```

```
Database logins with no expiration date pose a security threat.

Remote value:

"distributor_admin"

Policy value:

NULL
```

Cette sortie indique qu'aucune date d'expiration n'est configurée pour le compte « distributor_admin » qui doit être vérifié par rapport à la stratégie de sécurité du système.

Exemple 2 : Vérifier l'état activé d'une procédure mémorisée non autorisée

Cet audit vérifie si la procédure mémorisée « SQL Mail XPs » est activée. Les procédures mémorisées externes peuvent constituer une menace de sécurité pour certains systèmes et doivent souvent être désactivées.

```
<check_type: "Database" db_type:"SQLServer" version:"1">
<group_policy: "Unauthorized stored procedure check">
<custom_item>
 type: SQL_POLICY
 description: "SQL Mail XPs external stored procedure check"
 info: "Checking whether SQL Mail XPs is disabled."
 sql_request: "select value_in_use from sys.configurations where name = 'SQL Mail
       XPs'"
 sql_types: POLICY_INTEGER
 sql_expect: 0
</custom_item>
</group_policy
</check_type>
```

Le contrôle ci-dessus renverra un résultat de réussite (« passed ») si la procédure mémorisée « SQL Mail XPs » est désactivée (value_in_use = 0). Sinon, elle renverra un résultat d'échec (« failed »).

Exemple 3 : Vérifier l'état de la base de données avec des sql_types de résultat mixte

Dans certains cas, les interrogations de base de données de conformité exigent plusieurs demandes de données avec plusieurs résultats de type de données. L'exemple de vérification ci-dessous mélange les types de données et démontre comment la sortie peut être analysée.

```
<check_type: "Database" db_type:"SQLServer" version:"1">
<group_policy: "Mixed result type check">
<custom_item>
 type: SQL_POLICY
 description: "Mixed result type check"
 info: "Checking values for the master database."
 sql_request: " select database_id,user_access_desc,is_read_only from sys.databases
       where is_trustworthy_on=0 and name = 'master'"
 sql_types: POLICY_INTEGER,POLICY_VARCHAR,POLICY_INTEGER
 sql_expect: 1,MULTI_USER,0
</custom_item>
</group_policy
</check_type>
```

Les valeurs **`sql_request`**, **`sql_types`** et **`sql_expect`** contiennent toutes des valeurs séparées par une virgule.

Conditions

Il est possible de définir la logique **`if`**/**`then`**/**`else`** dans la stratégie de base de données. Ceci permet à l'utilisateur final de renvoyer un message d'avertissement plutôt qu'un message de réussite/échec en cas de réussite de l'audit.

La syntaxe pour exécuter les conditions est la suivante :

```
<if>
 <condition type: "or">
  <Insert your audit here>
 </condition>
 <then>
  <Insert your audit here>
 </then>
 <else>
  <Insert your audit here>
 </else>
</if>
```

Exemple :

```
<if>
 <condition type: "or">
  <custom_item>
   type: SQL_POLICY
   description: "clr enabled option"
   info: "Is CLR enabled?"
   sql_request: "select value_in_use from sys.configurations where name = 'clr
       enabled'"
   sql_types: POLICY_INTEGER
   sql_expect: "0"
  </custom_item>
 </condition>

 <then>
  <custom_item>
   type: SQL_POLICY
   description: "clr enabled option"
   info: "CLR is disabled?"
   sql_request: "select value_in_use from sys.configurations where name = 'clr
       enabled'"
   sql_types: POLICY_INTEGER
   sql_expect: "0"
  </custom_item>
 </then>

 <else>
  <report type: "WARNING">
   description: "clr enabled option"
   info: "CLR(Command Language Runtime objects) is enabled"
   info: "Check system policy to confirm CLR requirements."
  </report>
 </else>
</if>
```

L'échec ou la réussite de la condition n'est jamais indiqué dans le rapport car il s'agit d'une vérification « muette ».

Les conditions peuvent être du type « **and** » ou « **or** ».

Référence des fichiers de conformité d'audit pour la configuration UNIX

Cette section décrit les fonctions intégrées des contrôles de conformité Unix et les raisons de chaque paramètre.

Emploi des guillemets :

Les guillemets simples et doubles sont interchangeables autour des champs d'audit, sauf dans les deux cas suivants :

1. Dans les contrôles de conformité Windows où des champs spéciaux tels que CRLF, doivent être interprétés littéralement, utilisez des guillemets simples. Tout champ intégré qui doit être interprété comme une chaîne doit être inclus dans une séquence d'échappement.

Par exemple :
```
expect: 'First line\r\nSecond line\r\nJohn\'s Line'
```

2. Des guillemets doubles sont requis lorsque vous utilisez les fichiers Windows « include_paths » et « exclude_paths ».

Si des chaînes sont utilisées dans tout type de champ (description, value_data, regex, etc.) contenant des guillemets simples ou doubles, il y a deux façons de les traiter :

a. Utilisez le type de guillemets opposés pour les guillemets extérieurs.

Par exemple :
```
expect: "This is John's Line"
expect: 'We are looking for a double-quote-".*'
```

b. Échappez tout guillemet intégré avec une barre oblique inverse (guillemets doubles uniquement).

Par exemple :
```
expect: "\"Text to be searched\""
```

Type de contrôle

Tous les contrôles de conformité Unix doivent être mis entre crochets avec l'encapsulation « **check_type** » et la désignation « Unix ». L'Annexe A contient un exemple de contrôle de conformité Unix commençant par le paramètre **check_type** pour « Unix » et se terminant par la balise « **</check_type>** ».

Ceci est requis afin de distinguer les fichiers **.audit** conçus pour les audits de conformité de Windows (ou autres plateformes).

Le fichier est lu via le protocole SSH dans une mémoire tampon sur le serveur Nessus, puis le tampon est traité afin de vérifier la conformité/non-conformité.

Mots clés

Le tableau suivant indique comment chaque mot clé peut être utilisé dans les contrôles de conformité Unix :

Mot clé	Exemple d'utilisation et paramètres pris en charge
attr	Ce mot clé est utilisé avec FILE_CHECK et FILE_CHECK_NOT pour vérifier les attributs de fichier associés à un fichier. Veuillez vous reporter à la page principale

	chattr(1) pour des détails concernant la configuration des attributs d'un fichier.
comment	Ce champ est utilisé pour ajouter toute information supplémentaire qui n'est pas intégrable au champ de description. Exemple : `comment: (CWD - Current working directory)`
description	Ce mot clé fournit une brève description du contrôle en cours. Il est nécessaire que le champ **description** soit unique et deux contrôles quelconques ne doivent pas avoir le même champ de description. SecurityCenter de Tenable utilise ce champ pour produire automatiquement un numéro d'ID de plugin unique basé sur le champ **description**. Exemple : `description: "Permission and ownership check for /etc/at.allow"`
dont_echo_cmd	Ce mot clé est utilisé avec les audits de contrôle de conformité Unix « CMD_EXEC » et il indique à l'audit d'omettre l'exécution de la commande par le contrôle à partir de la sortie. Seuls les résultats de la commande sont affichés. Exemple : `dont_echo_cmd: YES`

except	Ce mot clé est utilisé pour exclure certains utilisateurs, services et fichiers du contrôle.
	
	Exemple :
	`except: "guest"`
	
	Plusieurs comptes d'utilisateurs peuvent être regroupés.
	
	Exemple :
	`except: "guest" | "guest1" | "guest2"`

expect	Ce mot clé est utilisé avec **regex**. Il permet de rechercher des valeurs spécifiques dans les fichiers.
	
	Exemple :
	`<custom_item>`
	` system: "Linux"`
	` type: FILE_CONTENT_CHECK`
	` description: "This check reports a problem when the log level setting in the sendmail.cf file is less than the value set in your security policy."`
	` file: "sendmail.cf"`
	` regex: ".*LogLevel=.*"`
	` expect: ".*LogLevel=9"`
	
	`</custom_item>`

file	Ce mot clé est utilisé pour décrire le chemin absolu ou relatif du fichier dont vous voulez vérifier les permissions et les paramètres de propriété.
	
	Exemples :
	`file: "/etc/inet/inetd.conf"`
	`file: "~/inetd.conf"`
	
	La valeur **file** peut aussi être un « glob »

	Exemple : `file: "/var/log/*"` Cette fonction est particulièrement utile lorsque vous voulez vérifier les permissions ou le contenu de tous les fichiers d'un répertoire donné au moyen de FILE_CHECK, FILE_CONTENT_CHECK, FILE_CHECK_NOT ou FILE_CONTENT_CHECK_NOT.

file_type	Ce mot clé décrit le type de fichier qui est recherché. La liste des types de fichier pris en charge est fournie ci-dessous.
	
	• b - bloc (tamponné) spécial
	• c - caractère (non tamponné) spécial
	• d - répertoire
	• p - canal nommé (FIFO)
	• f - fichier normal
	
	Exemple :
	`file_type: "f"`
	
	Un ou plusieurs types de fichier peuvent être transmis ensemble dans une même chaîne.
	
	Exemple :
	`file_type: "c|b"`

group	Ce mot clé est utilisé pour spécifier le groupe d'un fichier ; il est toujours utilisé avec le mot clé **file**. Le mot clé **group** peut avoir la valeur « none » (aucun), qui facilite les recherches de fichier sans propriétaire.
	
	Exemple :
	`group: "root"`
	
	Le groupe peut aussi être spécifié avec une condition logique « OR » (OU) en utilisant la syntaxe suivante :
	
	`group: "root" || "bin" || "sys"`

ignore	Ce mot clé indique au contrôle d'ignorer les fichiers désignés dans la recherche. Ce mot clé est disponible pour les types de vérification FILE_CHECK, FILE_CHECK_NOT, FILE_CONTENT_CHECK et FILE_CONTENT_CHECK_NOT.
	
	Exemples :
	`# ignore single file`
	`ignore: "/root/test/2"`
	
	`# ignore certain files from a directory`
	`ignore: "/root/test/foo*"`
	
	`# ignore all files in a directory`
	`ignore: "/root/test/*"`

info	Ce mot clé est utilisé pour ajouter une description plus détaillée du contrôle en cours telle qu'une réglementation, une URL, une stratégie d'entreprise ou une autre raison pour laquelle le paramètre est requis. Plusieurs champs **info** peuvent être ajoutés sur des lignes séparées pour formater le texte sous forme de paragraphe. Il n'existe pas de limite prédéfinie pour le nombre de champs **info** qui peuvent être utilisés.

	Exemple : `info: "ref. CIS_AIX_Benchmark_v1.0.1.pdf ch 1, pg 28-29."`
levels	Ce mot clé est utilisé avec CHKCONFIG, et il est utilisé pour spécifier les niveaux d'exécution pour lesquels un service doit être exploité. Tous les niveaux d'exécution doivent être décrits dans une seule chaîne. Par exemple, si le service « sendmail » (envoyer courrier) doit être exécuté aux niveaux d'exécution 1, 2 et 3, la valeur **levels** correspondante dans la vérification CHKCONFIG est : `levels: "123"`
mask	Ce mot clé est l'opposé du **mode** où vous pouvez spécifier des permissions qui **ne doivent pas** être disponibles pour un utilisateur, un groupe ou un autre membre particulier. Contrairement à **mode** qui recherche une valeur de permission *exacte*, les vérifications **mask** sont plus générales et vérifient si un fichier ou un répertoire possède un niveau de sécurisation égal ou supérieur à celui spécifié par **mask**. (Lorsque **mode** peut faire échouer un fichier avec une permission de 640 parce qu'il ne correspond pas à un audit qui attend une valeur de 644, **mask** verra que 640 est « plus sécurisé » et l'audit sera réussi.) Exemple : `mask: 022` Ceci spécifierait que toute permission est OK pour le propriétaire et exclurait les permissions d'écriture pour le groupe et autres membres. Une valeur **mask** de « 7 » signifierait qu'aucune permission n'est accordée pour ce propriétaire, ce groupe ou autre membre particulier.

md5	Ce mot clé est utilisé dans FILE_CHECK et FILE_CHECK_NOT pour assurer que le MD5 d'un fichier est en fait paramétré sur les conditions établies par la stratégie.
	
	Exemple :
	`<custom_item>`
	` type: FILE_CHECK`
	` description: "/etc/passwd has the proper md5 set"`
	` required: YES`
	` file: "/etc/passwd"`
	` md5: "ce35dc081fd848763cab2cfd442f8c22"`
	`</custom_item>`

mode	Ce mot clé décrit l'ensemble des permissions pour un fichier/dossier examiné. Le mot clé **mode** peut être représenté en format de chaîne ou en format octal.
	
	Exemples :
	`mode: "-rw-r--r--"`
	`mode: "644"`
	`mode: "7644"`

name	Ce mot clé est utilisé pour identifier le nom du processus dans PROCESS_CHECK. Exemple : `name: "syslogd"`
operator	Ce mot clé est utilisé avec RPM_CHECK et PKG_CHECK pour spécifier la condition qui détermine la réussite ou l'échec à un contrôle en fonction de la version du progiciel RPM installé. Il peut prendre les valeurs suivantes :

	• **lt** (inférieur à) • **lte** (inférieur ou égal à) • **gte** (supérieur ou égal à) • **gt** (supérieur à) • **eq** (égal à) Exemple : `operator: "lt"`

owner	Ce mot clé est utilisé pour spécifier le propriétaire d'un fichier ; il est toujours utilisé avec le mot clé **file**. Le mot clé **owner** peut avoir la valeur « none » (aucun), qui facilite les recherches de fichier sans propriétaire.
	
	Exemple :
	`owner: "root"`
	
	L'appartenance peut aussi être spécifiée avec une condition logique « OR » en utilisant la syntaxe suivante :
	`owner: "root" || "bin" || "adm"`

reference	Ce mot clé permet d'inclure des références croisées dans le fichier **.audit**. Il respecte le format « ref|ref-id1,ref|ref-id2 ». Exemple : `reference: "CAT|CAT II,800-53|IA-5,8500.2|IAIA-1,8500.2|IAIA-2,8500.2|IATS-1,8500.2|IATS-2"`
regex	Ce mot clé permet de chercher un fichier pour la correspondance à une expression regex particulière. Exemple : `regex: ".*LogLevel=9$"` Les caractères génériques suivants nécessitent un traitement spécial : + \ * ( ) ^ Échapper deux fois ces caractères avec deux barres obliques inverses « \\ » ou les placer entre crochets « [] » s'ils doivent être interprétés littéralement. D'autres caractères tels que ceux indiqués ci-dessous nécessitent seulement une barre oblique inverse pour être interprétés littéralement : . ? " ' La raison provient de la façon dont le compilateur traite ces caractères.
required	Ce mot clé est utilisé pour spécifier si l'élément vérifié doit être présent sur le système distant. Par exemple, lorsque **required** est paramétré sur « NO » (Non) et que le contrôle **type** sur « FILE_CHECK », le contrôle réussit si le fichier existe et que les permissions sont telles que spécifiées dans le fichier **.audit** ou si le fichier n'existe pas. D'un autre côté, si **required** est paramétré sur « YES » (Oui), le contrôle ci-dessus échoue.

rpm	Ce mot clé est utilisé pour spécifier le RPM à rechercher lorsqu'il est utilisé avec RPM_CHECK.
	
	Exemple :
	`<custom_item>`
	`type: RPM_CHECK`

	description: "Make sure that the Linux kernel is BELOW version 2.6.0" rpm: "kernel-2.6.0-0" operator: "lt" required: YES </custom_item>
search_locations	Ce mot clé peut être utilisé pour spécifier les emplacements qui font l'objet de la recherche dans un système de fichiers. Exemple : search_locations: "/bin" Plusieurs emplacements de recherche peuvent être regroupés ensemble. Exemple : search_locations: "/bin" | "/etc/init.d" | "/etc/rc0.d"
see_also	Ce mot clé permet d'inclure des liens à une référence. Exemple : see_also: "https://benchmarks.cisecurity.org/tools2/linux/CIS_Redhat_Linux_5_Benchmark_v2.0.0.pdf"
service	Ce mot clé est utilisé avec CHKCONFIG, XINETD_SVC et SVC_PROP, et il est utilisé pour spécifier le service vérifié. Exemple : <custom_item> type: CHKCONFIG description: "2.1 Disable Standard Services – Check if cups is disabled" service: "cups" levels: "123456" status: OFF </custom_item>
severity	Dans tout test, **<item>** ou **<custom_item>**, un indicateur « **severity** » peut être ajouté et paramétré sur « LOW » (Faible), « MEDIUM » (Moyen) ou « HIGH » (Élevé). Par défaut, les résultats non conformes sont indiqués comme « high ». Exemple : severity: MEDIUM
solution	Ce mot clé permet d'inclure un texte « Solution », le cas échéant. Exemple : solution: "Remove this file, if its not required"
status	Ce mot clé est utilisé dans PROCESS_CHECK, CHKCONFIG et XINETD_SVC pour déterminer si un service exécuté sur un hôte donné devrait être exécuté ou désactivé. Le mot clé **status** peut prendre 2 valeurs : « ON » (Activé) ou « OFF » (Désactivé). Exemple : status: ON status: OFF

system	Ce mot clé spécifie le type de système sur lequel le contrôle doit être effectué.  Le mot clé « **system** » est seulement applicable aux vérifications « **custom_item** », mais pas aux contrôles « **item** » intégrés. Les valeurs disponibles sont celles renvoyées par la commande « **uname** » sur le système d'exploitation cible. Par exemple, la valeur est « SunOS » sur Solaris, « Darwin » sur Mac OS X, « FreeBSD » sur FreeBSD, etc. Exemple : `system: "SunOS"`
timeout	Ce mot clé est utilisé avec CMD_EXEC et il spécifie, en secondes, la période de temps pendant laquelle la commande spécifiée sera autorisée à être exécutée avant sa temporisation. Ce mot clé est utile dans les cas où une commande particulière, telle qu'une commande Unix « **find** », nécessite une exécution prolongée. Si ce mot clé n'est pas spécifié, la temporisation par défaut pour les vérifications CMD_EXEC est cinq minutes. Exemple : `timeout: "600"`

type	`CHKCONFIG`
	`CMD_EXEC`
	`FILE_CHECK`
	`FILE_CHECK_NOT`
	`FILE_CONTENT_CHECK`
	`FILE_CONTENT_CHECK_NOT`
	`GRAMMAR_CHECK`
	`PKG_CHECK`
	`PROCESS_CHECK`
	`RPM_CHECK`
	`SVC_PROP`
	`XINETD_SVC`

value	Le mot clé **value** est utile pour vérifier si un paramètre du système confirme la valeur de la stratégie. Exemple : `value: "90..max"` Le mot clé **value** peut être spécifié comme plage [nombre..max]. Si la valeur se trouve entre le nombre spécifié et « max », la vérification réussit.

Éléments personnalisés

Un élément personnalisé est un contrôle complet défini en fonction des mots clés ci-dessus. Une liste d'éléments personnalisés est indiquée ci-dessous. Chaque contrôle commence par une balise « **<custom_item>** » et se termine par « **</custom_item>** ». Les balises contiennent des listes d'un ou plusieurs mots clés qui sont interprétés par l'analyseur syntaxique de contrôle de conformité pour effectuer les contrôles.

Les contrôles d'audit personnalisé peuvent utiliser indifféremment « **</custom_item>** » et « **</item>** » pour la balise de fermeture.

AUDIT_XML

Le contrôle d'audit « AUDIT_XML » permet d'examiner et de vérifier le contenu d'un fichier XML en appliquant les transformations XSL et en extrayant les données pertinentes, puis de déterminer la conformité à partir des mots clés **`regex`**, **`expect`**, et **`not_expect`** (voir l'Annexe C pour plus d'informations). Ce contrôle comprend au moins quatre mots clés, un type de mots clé, un fichier de description et des directives xsl_stmt (obligatoires), suivis par les mots clés **`regex`**, **`expect`** ou **`not_expect`** pour vérifier le contenu.

Exemple :

```
<custom_item>
 type: AUDIT_XML
 description: "1.14 - Ensure Oracle Database persistence plugin is set correctly -
       'DatabasePersistencePlugin'"
 file: "/opt/jboss-5.0.1.GA/server/all/deploy/ejb2-timer-service.xml"
 xsl_stmt: "<xsl:template match=\"server\">"
 xsl_stmt: "DatabasePersistencePlugin = <xsl:value-of
       select=\"/server/mbean[@code='org.jboss.ejb.txtimer.DatabasePersistencePolicy']/
       attribute[@name='DatabasePersistencePlugin']/text()\"/>"
 xsl_stmt: "</xsl:template>"
 regex: "DatabasePersistencePlugin = .+"
 not_expect: "org.jboss.ejb.txtimer.GeneralPurposeDatabasePersistencePlugin"
</custom_item>
```

Le mot clé file accepte les caractères génériques. Par exemple :

```
<custom_item>
 type: AUDIT XML
 description: "1.14 - Ensure Oracle Database persistence plugin is set correctly -
       'DatabasePersistencePlugin'"
 file: "/opt/jboss-5.0.1.GA/server/all/deploy/ejb2-*.xml"
 xsl_stmt: "<xsl:template match=\"server\">"
 xsl stmt: "DatabasePersistencePlugin = <xsl:value-of
       select=\"/server/mbean[@code='org.jboss.ejb.txtimer.DatabasePersistencePolicy']/
       attribute[@name='DatabasePersistencePlugin']/text()\"/>"
 xsl_stmt: "</xsl:template>"
 regex: "DatabasePersistencePlugin = .+"
 not expect: "org.jboss.ejb.txtimer.GeneralPurposeDatabasePersistencePlugin"
</custom_item>
```

CHKCONFIG

Le contrôle d'audit « CHKCONFIG » permet l'interaction avec l'utilitaire « **`chkconfig`** » sur le système distant Red Hat vérifié. Ce contrôle comporte cinq mots clés obligatoires : **`type`**, **`description`**, **`service`**, **`levels`** et **`status`**.

L'audit CHKCONFIG fonctionne uniquement sur les systèmes Red Hat ou un dérivé d'un système Red Hat tel que Fedora.

Exemple :

```
<custom_item>
 type: CHKCONFIG
 description: "Make sure that xinetd is disabled"
 service: "xinetd"
```

```
  levels: "123456"
  status: OFF
</custom_item>
```

CMD_EXEC

Il est possible d'exécuter des commandes sur l'hôte distant et de vérifier que la sortie est conforme à la sortie attendue. Ce type de contrôle doit être utilisé avec précaution car il n'est pas toujours portable dans les différentes variantes d'Unix.

Le mot clé **`quiet`** indique à Nessus de **ne pas** montrer la sortie de la commande qui a échoué. Il peut être paramétré sur « YES » (Oui) ou « NO » (Non). Par défaut, il est paramétré sur « NO » (Non) et le résultat de la commande est affiché. De même, le mot clé « **`dont_echo_cmd`** » limite les résultats en produisant les résultats de la commande mais pas la commande elle-même.

Le mot clé **`nosudo`** permet à l'utilisateur de signifier à Nessus de **ne pas** utiliser sudo pour exécuter la commande en le paramétrant sur « YES » (Oui). Par défaut, il est paramétré sur « NO » (Non) et sudo est toujours utilisé lorsqu'il est configuré pour cela.

Exemple :

```
<custom_item>
 type: CMD_EXEC
 description: "Make sure that we are running FreeBSD 4.9 or higher"
 cmd: "uname -a"
 timeout: 7200
 expect: "FreeBSD (4\.(9|[1-9][0-9])|[5-9]\.)"
 dont_echo_cmd: YES
</custom_item>
```

FILE_CHECK

Les audits de conformité Unix mettent généralement à l'essai l'existence et les paramètres d'un fichier donné. L'audit « FILE_CHECK » utilise quatre mots clés ou plus pour permettre de spécifier ces contrôles. Les mots clés **`type`**, **`description`** et **`file`** sont obligatoires et sont suivis d'une ou plusieurs vérifications. La syntaxe actuelle prend en charge les contrôles pour les permissions de propriétaire, groupe et fichier.

Il est possible d'utiliser des globs dans FILE_CHECK (par exemple **`/var/log/*`**). Toutefois, il faut noter que les globs seront seulement étendus à des fichiers, et non à des répertoires. Si un glob est spécifié et qu'un ou plusieurs fichiers correspondants doivent être ignorés de la recherche, utilisez le mot clé « **`ignore`** » pour spécifier les fichiers à ignorer.

Les mots clés autorisés sont :

```
uid : ID d'utilisateur numérique (par exemple, 0)
gid : ID de groupe numérique (par exemple, 500)
check_uneveness : YES
system : Type de système (par exemple, Linux)
description : Description textuelle du contrôle de fichier
file : Chemin complet et fichier à vérifier (par exemple, /etc/sysconfig/sendmail)
owner : Propriétaire du fichier (par exemple, root)
group : Propriétaire de groupe du fichier (par exemple, bin)
mode : Mode d'autorisation (par exemple, 644)
mask : umask de fichier (par exemple, 133)
md5 : Empreinte numérique MD5 d'un fichier (par exemple,
88d3dbe3760775a00b900a850b170fcd)
ignore : Fichier à ignorer (par exemple, /var/log/secure)
attr : Attribut de fichier (par exemple, ----i--------)
```

Les permissions de fichier sont considérées comme inégales si les options « group » ou « other » sont dotées de permissions supplémentaires par rapport à « owner », ou si « other » est doté de permissions supplémentaires par rapport à « group ».

Voici quelques exemples :

```
<custom_item>
 system: "Linux"
 type: FILE_CHECK
 description: "Permission and ownership check for /etc/default/cron"
 file: "/etc/default/cron"
 owner: "bin"
 group: "bin"
 mode: "-r--r--r--"
</custom_item>
```

```
<custom_item>
 system: "Linux"
 type: FILE_CHECK
 description: "Permission and ownership check for /etc/default/cron"
 file: "/etc/default/cron"
 owner: "bin"
 group: "bin"
 mode: "444"
</custom_item>
```

```
<custom_item>
 system: "Linux"
 type: FILE_CHECK
 description: "Make sure /tmp has its sticky bit set"
 file: "/tmp"
 mode: "1000"
</custom_item>
```

```
<custom_item>
 type: FILE_CHECK
 description: "/etc/passwd has the proper md5 set"
 required: YES
 file: "/etc/passwd"
 md5: "ce35dc081fd848763cab2cfd442f8c22"
</custom_item>
```

```
<custom_item>
 type: FILE_CHECK
 description: "Ignore maillog in the file mode check"
 required: YES
 file: "/var/log/m*"
 mode: "1000"
 ignore: "/var/log/maillog"
</custom_item>
```

FILE_CHECK_NOT

La vérification « FILE_CHECK_NOT » comprend au moins trois mots clés. Les mots clés **type**, **description** et **file** sont obligatoires et sont suivis d'une ou plusieurs vérifications. La syntaxe actuelle prend en charge les contrôles pour les permissions de propriétaire, groupe et fichier. À l'instar de la vérification FILE_CHECK, le mot clé « **ignore** » peut être utilisé pour ignorer un ou plusieurs fichiers si un glob de fichier est spécifié.

Cette fonction est l'inverse de FILE_CHECK. Une stratégie échoue si un fichier n'existe pas ou si son mode est le même que celui défini dans le contrôle lui-même.

Il est possible d'utiliser des globs dans FILE_CHECK_NOT (par exemple **/var/log/***). Toutefois, il faut noter que les globs seront seulement étendus à des fichiers, et non à des répertoires.

Voici quelques exemples :

```
<custom item>
 type: FILE_CHECK_NOT
 description: "Make sure /bin/bash does NOT belong to root"
 file: "/bin/bash"
 owner: "root"
</custom_item>
```

```
<custom_item>
 type: FILE_CHECK_NOT
 description: "Make sure that /usr/bin/ssh does NOT exist"
 file: "/usr/bin/ssh"
</custom_item>
```

```
<custom_item>
 type: FILE_CHECK_NOT
 description: "Make sure /root is NOT world writeable"
 file: "/root"
 mode: "0777"
</custom_item>
```

FILE_CONTENT_CHECK

Comme pour le test de l'existence et des paramètres d'un fichier, le contenu des fichiers de texte peut aussi être analysé. Des expressions rationnelles peuvent être utilisées pour chercher un ou plusieurs emplacements pour un contenu existant. Utilisez le mot clé « **ignore** » pour ignorer un ou plusieurs fichiers de l'emplacement ou des emplacements de recherche spécifiés.

Le champ **string_required** peut être configuré afin de spécifier si la chaîne contrôlée qui fait l'objet de la recherche doit ou non être présente. Si cette option n'est pas définie, on suppose qu'elle est requise. Le champ **file_required** peut être configuré afin de spécifier si le fichier contrôlé doit ou non être présent. Si cette option n'est pas définie, on suppose qu'elle est requise.

Voici quelques exemples :

```
<custom_item>
 system: "Linux"
 type: FILE_CONTENT_CHECK
 description: "This check reports a problem when the log level setting in the
       sendmail.cf file is less than the value set in your security policy."
 file: "sendmail.cf"
```

```
 regex: ".*LogLevel=.*$"
 expect: ".*LogLevel=9"
</custom_item>
```

```
<custom_item>
 system: "Linux"
 type: FILE_CONTENT_CHECK
 file: "sendmail.cf"
 search_locations: "/etc:/etc/mail:/usr/local/etc/mail/"
 regex: ".*PrivacyOptions=".*"
 expect: ".*PrivacyOptions=.*,novrfy,.*"
</custom_item>
```

```
<custom_item>
 #System: "Linux"
 type: FILE_CONTENT_CHECK
 description: "FILE_CONTENT_CHECK"
 file: "/root/test2/foo*"
 # ignore single file
 ignore: "/root/test/2"
 # ignore all files in a directory
 ignore: "/root/test/*"
 #ignore certain files from a directory
 ignore: "/root/test/foo*"
 regex: "FOO"
 expect: "FOO1"
 file_required: NO
 string_required: NO
</custom_item>
```

En ajoutant un « ~ » à un paramètre de fichier, FILE_CONTENT_CHECK peut scanner les répertoires d'accueil de l'utilisateur pour détecter le contenu non conforme.

```
<custom_item>
 system: "Linux"
 type: FILE_CONTENT_CHECK
 description: "Check all user home directories"
 file: "~/.rhosts"
 ignore: "/.foo"
 regex: "\\+"
 expect: "\\+"
</custom_item>
```

FILE_CONTENT_CHECK_NOT

Cet audit examine le contenu d'un fichier pour une correspondance avec la description regex dans le champ **`regex`**. Cette fonction inverse FILE_CONTENT_CHECK. Autrement dit, une stratégie échoue si la regex correspond **réellement** dans le fichier. Utilisez le mot clé « **`ignore`** » pour ignorer un ou plusieurs fichiers de l'emplacement ou des emplacements de recherche spécifiés.

Cet élément de stratégie vérifie si le fichier contient l'expression rationnelle regex et si cette expression ne correspond pas à **`expect`**.

Le type autorisé est :

```
value_type: POLICY_TEXT
value_data: "PATH\Filename"
regex: "regex"
expect: "regex"
```

Ce contrôle doit spécifier à la fois **regex** et **expect**.

Voici un exemple :

```
<custom_item>
 type: FILE_CONTENT_CHECK_NOT
 description: "Make sure NIS is not enabled on the remote host by making sure that
       '+::' is not in /etc/passwd"
 file: "/etc/passwd"
 regex: "^\+::"
 expect: "^\+::"
 file_required: NO
 string_required: NO
</custom_item>
```

GRAMMAR_CHECK

Le contrôle d'audit « GRAMMAR_CHECK » examine le contenu d'un fichier et fait correspondre une grammaire librement définie (constituée d'une ou plusieurs déclarations regex). Si **une** ligne du fichier cible ne correspond à aucune des déclarations regex, le test échoue.

Exemple :

```
<custom_item>
 type: GRAMMAR_CHECK
 description: "Check /etc/securetty contents are OK."
 file: "/etc/securetty"
 regex: "console"
 regex: "vc/1"
 regex: "vc/2"
 regex: "vc/3"
 regex: "vc/4"
 regex: "vc/5"
 regex: "vc/6"
 regex: "vc/7"
</custom_item>
```

MACOSX_DEFAULTS_READ

Le contrôle d'audit « MACOSX_DEFAULTS_READ » examine les valeurs système par défaut sur MAC OS X. Le comportement de ce contrôle varie suivant les propriétés définies.

Si la propriété **plist_user** est définie sur « all », tous les paramètres utilisateurs sont vérifiés. Sinon, seul le paramètre utilisateur spécifié est vérifié.

Si la propriété **byhost** est définie sur `YES` alors que la propriété **plist_user** est définie, l'interrogation suivante est exécutée :

```
/usr/bin/defaults -currentHost read /Users/foo/Library/Preferences/ByHost/plist_name
       plist_item
```

Si la propriété **byhost** n'est pas définie (alors que la propriété **plist_user** est définie), l'interrogation suivante est exécutée :

```
/usr/bin/defaults -currentHost read /Users/foo/Library/Preferences/plist_name
      plist_item
```

Si la propriété **byhost** n'est pas définie (alors que la propriété **plist_user** n'est pas définie), l'interrogation suivante est exécutée :

```
/usr/bin/defaults -currentHost read plist_name plist_item
```

Les propriétés suivantes sont prises en charge :

plist_name : l'élément plist que nous voulons interroger, par exemple com.apple.digihub
plist_item : l'élément plist à vérifier, par exemple com.apple.digihub.blank.cd.appeared
plist_option : CANNOT_BE_NULL. Lorsque cette propriété est définie sur CANNOT_BE_NULL, la vérification échoue si le paramètre à vérifier n'est pas défini.
byhost : YES. Si la propriété byhost est définie sur YES (Oui), l'interrogation sera légèrement différente.

Exemples :

```
<custom_item>
 system: "Darwin"
 type: MACOSX_DEFAULTS_READ
 description: "Automatic actions must be disabled for blank CDs - 'action=1;'"
 plist_user: "all"
 plist_name: "com.apple.digihub"
 plist_item: "com.apple.digihub.blank.cd.appeared"
 regex: "\\s*action\\s*=\\s*1;"
 plist_option: CANNOT_BE_NULL
</custom_item>


<custom_item>
 system: "Darwin"
 type: MACOSX_DEFAULTS_READ
 description: "System must have a password-protected screen saver configured to DoD"
 plist_user: "all"
 plist_name: "com.apple.screensaver"
 byhost: YES
 plist_item: "idleTime"
 regex: "[A-Za-z0-9_-]+\\s*=\\s*(900|[2-8][0-9][0-9]|1[8-9][0-9])$"
 plist_option: CANNOT_BE_NULL
</custom_item>

<custom_item>
 system: "Darwin"
 type: MACOSX_DEFAULTS_READ
 description: "System must have a password-protected screen saver configured to DoD"
 plist_name: "com.apple.screensaver"
 plist_item: "idleTime"
 regex: "[A-Za-z0-9_-]+\\s*=\\s*(900|[2-8][0-9][0-9]|1[8-9][0-9])$"
 plist_option: CANNOT_BE_NULL
</custom_item>
```

PKG_CHECK

Le contrôle d'audit « PKG_CHECK » effectue un **pkgchk** sur un système SunOS. Le mot clé **pkg** est utilisé pour spécifier le progiciel à rechercher et le mot clé **operator** spécifie la condition de réussite ou d'échec du contrôle en fonction de la version du progiciel installée.

Exemples :

```
<custom_item>
 system: "SunOS"
 type: PKG_CHECK
 description: "Make sure SUNWcrman is installed"
 pkg: "SUNWcrman"
 required: YES
</custom_item>
```

```
<custom_item>
 system: "SunOS"
 type: PKG_CHECK
 description: "Make sure SUNWcrman is installed and is greater than 9.0.2"
 pkg: "SUNWcrman"
 version: "9.0.2"
 operator: "gt"
 required: YES
</custom_item>
```

PROCESS_CHECK

Comme pour les contrôles de fichier, une plateforme Unix vérifiée peut être testée pour vérifier ses processus en cours d'exécution. Cette fonction exécute la commande « **chkconfig –list** » pour obtenir une liste de processus exécutés.

Exemples :

```
<custom_item>
 system: "Linux"
 type: PROCESS_CHECK
 name: "auditd"
 status: OFF
</custom_item>
```

```
<custom_item>
 system: "Linux"
 type: PROCESS_CHECK
 name: "syslogd"
 status: ON
</custom_item>
```

RPM_CHECK

Le contrôle d'audit « RPM_CHECK » est utilisé pour vérifier les numéros de version des progiciels RPM installés sur le système distant. Ce contrôle comprend quatre mots clés obligatoires : **type**, **description**, **rpm**, **operator**, et un mot clé facultatif **required**. Le mot clé **rpm** est utilisé pour spécifier le progiciel à rechercher et le mot clé **operator** spécifie la condition de réussite ou d'échec du contrôle en fonction de la version du progiciel RPM installée.

L'utilisation des contrôles RPM n'est pas portable sur les distributions Linux. Par conséquent, l'utilisation de RPM_CHECK n'est pas considérée comme portable.

Voici quelques exemples, qui supposent que **`iproute-2.4.7-10`** est installé :

```
<custom_item>
 type: RPM_CHECK
 description: "RPM check for iproute-2.4.7-10 - should pass"
 rpm: "iproute-2.4.7-10"
 operator: "gte"
</custom_item>
```

```
<custom_item>
 type: RPM_CHECK
 description: "RPM check for iproute-2.4.7-10 should fail"
 rpm: "iproute-2.4.7-10"
 operator: "lt"
 required: YES
</custom_item>
```

```
<custom_item>
 type: RPM_CHECK
 description: "RPM check for iproute-2.4.7-10 should fail"
 rpm: "iproute-2.4.7-10"
 operator: "gt"
 required: NO
</custom_item>
```

```
<custom_item>
 type: RPM_CHECK
 description: "RPM check for iproute-2.4.7-10 should pass"
 rpm: "iproute-2.4.7-10"
 operator: "eq"
 required: NO
</custom_item>
```

SVC_PROP

Le contrôle d'audit « SVC_PROP » permet d'interagir avec l'outil « **`svcprop -p`** » sur un système Solaris 10. Ceci peut être utilisé pour interroger les propriétés associées à un service particulier. Le mot clé **`service`** est utilisé pour spécifier le service vérifié. Le mot clé **`property`** spécifie le nom de la propriété que nous souhaitons interroger. Le mot clé **`value`** est la valeur attendue de la propriété. La valeur attendue peut aussi être une regex.

Le champ **`svcprop_option`** peut être configuré afin de spécifier si la chaîne contrôlée qui fait l'objet de la recherche doit ou non être présente. Ce champ permet d'accéder aux arguments CAN_BE_NULL ou CANNOT_BE_NULL.

Exemples :

```
<custom_item>
 type: SVC_PROP
 description: "Check service status"
```

```
 service: "cde-ttdbserver:tcp"
 property: "general/enabled"
 value: "false"
</custom_item>
```

```
<custom_item>
 type: SVC_PROP
 description: "Make sure FTP logging is set"
 service: "svc:/network/frp:default"
 property: "inetd_start/exec"
 regex: ".*frpd.*-l"
</custom_item>
```

```
<custom_item>
 type: SVC_PROP
 description: "Check if ipfilter is enabled - can be missing or not found"
 service: "network/ipfilter:default"
 property: "general/enabled"
 value: "true"
svcprop_option: CAN_BE_NULL
</custom_item>
```

XINETD_SVC

Le contrôle d'audit « XINETD_SVC » est utilisé pour vérifier l'état de démarrage des services xinetd. La vérification comprend quatre mots clés obligatoires : **type**, **description**, **service** et **status**.

Ce contrôle fonctionne uniquement sur les systèmes Red Hat ou un dérivé d'un système Red Hat tel que Fedora.

Exemple :

```
<custom_item>
 type: XINETD_SVC
 description: "Make sure that telnet is disabled"
 service: "telnet"
 status: OFF
</custom_item>
```

Contrôles intégrés

Les contrôles qui ne pouvaient pas être couverts par les contrôles décrits ci-dessus doivent être écrits en tant que noms personnalisés dans NASL. Tous ces contrôles appartiennent à la catégorie des contrôles « intégrés ». Chaque contrôle commence par une balise « **<item>** » et se termine par « **</item>** ». Les balises contiennent des listes d'un ou plusieurs mots clés qui sont interprétés par l'analyseur syntaxique de contrôle de conformité pour effectuer les contrôles. Voici une liste des contrôles disponibles.

Le mot clé « **system** » n'est pas disponible pour les contrôles intégrés et son utilisation entraînera une erreur de syntaxe.

Gestion des mots de passe

Dans les exemples ci-dessous, <min> et <max> sont utilisés pour représenter une valeur entière, et non une chaîne, à utiliser dans les données de valeur d'audit.

Lorsque la valeur minimale ou maximale exacte n'est pas connue, remplacez la valeur entière par les chaînes « Min » ou « Max ».

min_password_length

Utilisation

```
<item>
 name: "min_password_length"
 description: "This check examines the system configuration for the minimum password
length that the passwd program will accept. The check reports a problem if the minimum
length is less than the length specified in your policy."
 except: "user1" | "user2" (list of users to be excluded)
 value: "<min>..<max>"
</item>
```

Cette vérification intégrée assure que la longueur minimale du mot de passe mise en œuvre sur le système à distance est dans la plage « **<min>..<max>** ». L'existence d'une longueur minimale du mot de passe force les utilisateurs à choisir des mots de passe plus complexes.

Système d'exploitation	Mise en œuvre
Linux	La longueur minimale du mot de passe est définie comme PASS_MIN_LEN dans **/etc/login.defs**.
Solaris	La longueur minimale du mot de passe est définie comme PASSLENGTH dans **/etc/default/passwd**. Ce paramètre contrôle aussi la longueur maximale du mot de passe.
HP-UX	La longueur minimale du mot de passe est définie comme MIN_PASSWORD_LENGTH dans **/etc/default/security**.
Mac OS X	La longueur minimale du mot de passe est définie comme « minChar » dans la stratégie locale, définie à l'aide de la commande **pwpolicy**.

Exemple :

```
<item>
 name: "min_password_length"
 description: "Make sure that each password has a minimum length of 6 chars or more"
 value: "6..65535"
</item>
```

max_password_age

Utilisation
<item> name: "max_password_age" description: "This check reports agents that have a system default maximum password age greater than the specified value and agents that do not have a maximum password age setting." except: "user1" | "user2" (list of users to be excluded) value: "<min>..<max>" </item>

Cette fonction intégrée assure que l'antériorité maximale du mot de passe (par exemple le moment où les utilisateurs doivent changer leur mot de passe) respecte la plage définie.

L'existence d'une antériorité maximale du mot de passe empêche les utilisateurs de conserver le même mot de passe pendant plusieurs années. Les changements de mot de passe aident souvent à empêcher une personne malveillante possédant un mot de passe de l'utiliser indéfiniment.

Système d'exploitation	Mise en œuvre
Linux	La variable PASS_MAX_DAYS est définie dans **/etc/login.defs**.
Solaris	La variable MAXWEEKS dans **/etc/default/passwd** définit le nombre maximal de semaines pendant lesquelles un mot de passe peut être utilisé.
HP-UX	Cette valeur est contrôlée par la variable PASSWORD_MAXDAYS dans **/etc/default/security**.
Mac OS X	L'option « maxMinutesUntilChangePassword » de la stratégie de mot de passe (telle qu'établie avec l'outil **pwpolicy**) peut être utilisée pour paramétrer cette valeur.

Exemple :

```
<item>
 name: "max_password_age"
 description: "Make sure a password can not be used for more than 21 days"
 value: "1..21"
</item>
```

min_password_age

Utilisation
<item> name: "min_password_age" description: "This check reports agents and users with password history settings that are less than a specified minimum number of passwords." except: "user1" | "user2" (list of users to be excluded) value: "<min>..<max>" </item>

Cette fonction intégrée assure que l'antériorité minimale du mot de passe (par exemple la période requise avant que les utilisateurs soient autorisés à changer leur mot de passe) respecte la plage définie.

L'existence d'une antériorité minimale du mot de passe empêche les utilisateurs de changer trop souvent de mot de passe pour essayer de contourner l'historique d'antériorité maximale du mot de passe. Certains utilisateurs procèdent ainsi pour recycler leur mot de passe initial, ce qui contourne les exigences de changement du mot de passe.

Système d'exploitation	Mise en œuvre
Linux	La variable PASS_MIN_DAYS est définie dans **`/etc/login.defs`**.
Solaris	La variable MINWEEKS dans **`/etc/default/passwd`** définit le nombre maximal de semaines pendant lesquelles un mot de passe peut être utilisé.
HP-UX	Cette valeur est contrôlée par la variable PASSWORD_MINDAYS dans **`/etc/default/security`**.
Mac OS X	Cette option n'est pas prise en charge.

Exemple :

```
 <item>7
  name: "min_password_age"
  description: "Make sure a password cannot be changed before 4 days while allowing the
        user to change at least after 21 days"
  value: "4..21"
 </item>
```

Accès racine
root_login_from_console

Utilisation

`<item>`
` name: "root_login_from_console"`
` description: "This check makes sure that root can only log in from the system console (not remotely)."`
`</item>`

Cette fonction intégrée assure que l'utilisateur « root » doit se connecter au système distant directement par l'intermédiaire de la console physique.

La raison pour ce contrôle est le fait que les bonnes pratiques administratives interdisent l'utilisation directe du compte root (racine) pour que l'accès puisse être tracé à une personne particulière. Utilisez à la place un compte utilisateur générique (membre du groupe wheel sur les systèmes BSD), puis utilisez « **`su`** » (ou **`sudo`**) pour élever les privilèges afin de permettre l'exécution des tâches administratives.

Système d'exploitation	Mise en œuvre
Linux et HP-UX	Assurez-vous que **`/etc/securetty`** existe et contient seulement « console ».
Solaris	Assurez-vous que **`/etc/default/login`** contient la ligne « `CONSOLE=/dev/console` ».
Mac OS X	Cette option n'est pas prise en charge.

Gestion des permissions

accounts_bad_home_permissions

Utilisation

`<item>`
` name: "accounts_bad_home_permissions"`
` description: "This check reports user accounts that have home directories with incorrect user or group ownerships."`
`</item>`

Cette fonction intégrée assure que le répertoire d'accueil de chaque utilisateur non privilégié appartient à l'utilisateur et que des utilisateurs tiers (appartenant au même groupe ou « everyone ») ne peuvent pas y écrire. Il est généralement recommandé que les répertoires d'accueil d'un utilisateur soient paramétrés en mode 0755 ou à un mode plus restrictif (par exemple 0700). Ce test réussit si chaque répertoire d'accueil est configuré correctement et échoue dans le cas contraire. L'un des deux mots clés **`mode`** ou **`mask`** peut être utilisé ici pour spécifier les niveaux de permission souhaités pour les répertoires d'accueil. Le mot clé **`mode`** accepte les répertoires d'accueil correspondant exactement à un niveau spécifié et le mot clé **`mask`** accepte les répertoires d'accueil qui possèdent le niveau de sécurisation spécifié ou un niveau plus élevé.

Si des tiers peuvent écrire dans le répertoire d'accueil d'un utilisateur, ils peuvent forcer l'utilisateur à exécuter des commandes arbitraires en altérant les fichiers **`~/.profile`**, **`~/.cshrc`**, **`~/.bashrc`**.

Si des fichiers doivent être partagés entre des utilisateurs du même groupe, il est en général recommandé d'utiliser un répertoire inscriptible dédié au groupe, et non le répertoire d'accueil d'un utilisateur.

Pour tout répertoire d'accueil mal configuré, exécutez « **`chmod 0755 <user directory>`** » et changez l'appartenance en conséquence.

accounts_bad_home_group_permissions

Utilisation

`<item>`
` name: "accounts_bad_home_group_permissions"`
` description: "This check makes sure user home directories are group owned by the user's primary group."`
`</item>`

Sur le plan opérationnel, cette fonction intégrée est similaire à **`accounts_bad_home_permissions`**, mais elle garantit que les répertoires d'accueil de l'utilisateur sont la propriété d'un groupe : le groupe principal de l'utilisateur.

accounts_without_home_dir

Utilisation

```
<item>
 name: "accounts_without_home_dir"
 description: "This check reports user accounts that do not have home directories."
</item>
```

Cette fonction intégrée assure que chaque utilisateur a un répertoire d'accueil. Elle réussit si un répertoire valide est attribué à chaque utilisateur et échoue dans le cas contraire. L'appartenance ou les permissions de répertoire d'accueil ne sont pas vérifiées par ce contrôle.

Il est généralement recommandé que chacun des utilisateurs d'un système dispose d'un répertoire d'accueil défini car certains outils peuvent devoir le lire ou y écrire (par exemple, **`sendmail`** recherche un fichier **`~/.forward`**). Si un utilisateur n'a pas besoin de se connecter, un shell non existant (par exemple **`/bin/false`**) doit être défini. Sur de nombreux systèmes, un utilisateur sans répertoire d'accueil obtiendra quand même des privilèges de connexion mais son répertoire d'accueil actuel est /.

invalid_login_shells

Utilisation

```
<item>
 name: "invalid_login_shells"
 description: "This check reports user accounts with shells which do not exist or is not
listed in /etc/shells."
</item>
```

Cette fonction intégrée assure que chaque utilisateur a un shell valide tel que défini dans **`/etc/shells`**.

Le fichier **`/etc/shells`** est utilisé par des applications telles que Sendmail et les serveurs FTP pour déterminer si un shell est valide sur le système. Bien qu'il ne soit pas utilisé par le programme de connexion, les administrateurs peuvent utiliser ce fichier pour définir les shells qui sont valides sur le système. Le contrôle **`invalid_login_shells`** peut vérifier que tous les utilisateurs du fichier **`/etc/passwd`** sont configurés avec des shells valides tels que définis dans le fichier **`/etc/shells`**.

Ceci évite les pratiques non autorisées telles que l'utilisation de **`/sbin/passwd`** comme shell pour permettre aux utilisateurs de changer leur mot de passe. Si vous ne voulez pas qu'un utilisateur puisse se connecter, créez un shell non valide dans **`/etc/shells`** (par exemple « **`/nonexistent`** ») et définissez-la pour les utilisateurs souhaités.

Si aucun shell valide n'est défini pour certains utilisateurs, définissez-le pour eux.

login_shells_with_suid

Utilisation

```
<item>
 name: "login_shells_with_suid"
 description: "This check reports user accounts with login shells that have setuid or
setgid privileges."
</item>
```

Cette fonction intégrée assure qu'aucun shell n'a des capacités « set-uid ».

Un shell « setuid » signifie que, chaque fois que le shell démarre, le processus lui-même dispose des privilèges correspondants à ses permissions (par exemple, un shell setuid « root » accorde des privilèges de superutilisateur à tous les utilisateurs).

L'existence d'un shell « setuid » annule l'objectif d'avoir des UID et des GID et rend le contrôle de l'accès beaucoup plus complexe.

Supprimez le bit SUID de chaque shell « setuid ».

login_shells_writeable

Utilisation

`<item>`
` name: "login_shells_writeable"`
` description: "This check reports user accounts with login shells that have group or world write permissions."`
`</item>`

Cette fonction intégrée assure qu'aucun shell n'est inscriptible universellement/par le groupe.

Si un shell est inscriptible universellement (ou par le groupe), les utilisateurs non privilégiés peuvent la remplacer par un programme quelconque. Ceci permet à un utilisateur malveillant de forcer les autres utilisateurs de ce shell à exécuter des commandes arbitraires lorsqu'ils se connectent.

Assurez-vous que les permissions de chaque shell sont définies correctement.

login_shells_bad_owner

Utilisation

`<item>`
` name: "login_shells_bad_owner"`
` description: "This check reports user accounts with login shells that are not owned by root or bin."`
`</item>`

Cette fonction intégrée assure que chaque shell appartient aux utilisateurs « root » ou « bin ».

Comme pour les shells assortis de permissions non valides, si un utilisateur possède un shell utilisé par d'autres utilisateurs, il peut le modifier pour forcer les utilisateurs tiers à exécuter des commandes arbitraires lorsqu'ils se connectent.

Seuls les utilisateurs « root » et/ou les « bin » devraient être capables de modifier les binaires de l'ensemble du système.

Gestion du fichier de mots de passe

passwd_file_consistency

Utilisation

`<item>`
` name: "passwd_file_consistency"`
` description: "This check makes sure /etc/passwd is valid."`

```
</item>
```

Cette fonction intégrée assure que chaque ligne de **/etc/passwd** a un format valide (par exemple, sept champs séparés par deux-points). Si une ligne est mal formée, elle est signalée et le contrôle échoue.

La présence d'un fichier **/etc/passwd** mal formé peut rendre inutilisables plusieurs outils de gestion des utilisateurs. Elle peut aussi indiquer une intrusion ou un bug dans une application personnalisée de gestion des utilisateurs. Elle peut aussi montrer que quelqu'un a essayé d'ajouter un utilisateur avec un nom non valide (auparavant, il était habituel de créer un utilisateur appelé « toor:0:0 » pour obtenir des privilèges root).

Si le test est considéré non conforme, l'administrateur doit supprimer ou corriger les lignes illégales de **/etc/passwd**.

passwd_zero_uid

Utilisation

```
<item>
 name: "passwd_zero_uid"
 description: "This check makes sure that only ONE account has a uid of 0."
</item>
```

Cette fonction intégrée assure qu'un seul compte a un UID de « 0 » dans **/etc/passwd**. Elle doit normalement être réservée au compte « root » mais il est possible d'ajouter des comptes supplémentaires avec un UID de 0 qui auraient le même accès privilégié. Ce test réussit si un seul compte a un UID de 0 et échoue dans le cas contraire.

Un UID de « 0 » accorde des privilèges root sur le système. Un utilisateur root peut exécuter toutes les opérations voulues sur le système, en particulier surveiller l'utilisation de mémoire d'autres processus (ou du noyau), lire et écrire dans tout fichier sur le système, etc. Puisque ce compte est si privilégié, son utilisation doit être limitée au minimum et il doit être bien protégé.

Les bonnes pratiques administratives exigent que chaque UID soit unique (comme l'indique le « U » dans UID). L'existence de deux comptes (ou plus) avec des privilèges « root » annule la responsabilité qu'un administrateur de système peut avoir envers le système. En outre, beaucoup de systèmes limitent la connexion directe du root à la console seulement, pour que l'usage administratif puisse faire l'objet d'un suivi. Généralement, les administrateurs système doivent d'abord se connecter à leur propre compte et utiliser la commande **su** pour devenir root. Un compte supplémentaire avec un UID de 0 contourne cette restriction.

Si l'accès « root » doit être partagé entre les utilisateurs, utilisez plutôt un outil comme **sudo** ou **calife** (ou RBAC sur Solaris). Il ne doit exister qu'un seul compte avec un UID de « 0 ».

passwd_duplicate_uid

Utilisation

```
<item>
 name: "passwd_duplicate_uid"
 description: "This check makes sure that every UID in /etc/passwd is unique."
</item>
```

Cette fonction intégrée assure que chaque compte répertorié dans **/etc/passwd** a un UID unique. Ce test réussit si chaque UID est unique et échoue dans le cas contraire.

Sur un système Unix, chaque utilisateur est identifié par son UID (User ID - ID utilisateur), un nombre compris entre 0 et 65535. Si deux utilisateurs ont le même UID, non seulement ils reçoivent les mêmes privilèges, mais le système les considère comme une même personne. Ceci contourne toutes les responsabilités puisqu'il est impossible de déterminer quelles actions ont été effectuées par chaque utilisateur (généralement, le système effectuera une recherche en arrière sur l'UID et utilisera le premier nom des comptes partageant l'UID pour afficher les journaux).

Les normes de sécurité telles que les tests CIS interdisent de partager un UID entre utilisateurs. Si des utilisateurs doivent partager des fichiers, ils peuvent utiliser des groupes.

Donnez à chaque utilisateur du système un ID unique.

passwd_duplicate_gid

Utilisation

```
<item>
 name: "passwd_duplicate_gid"
 description: "This check makes sure that every GID in /etc/passwd is unique."
</item>
```

Cette fonction intégrée assure que l'ID de groupe primaire (GID) de chaque utilisateur est unique. Le test réussit si chaque utilisateur a un GID unique et échoue dans le cas contraire.

Les normes de sécurité recommandent de créer un groupe par utilisateur (généralement avec le même nom que le nom d'utilisateur). Dans cette configuration, les fichiers créés par l'utilisateur sont généralement « sécurisés par défaut » puisqu'ils appartiennent à son groupe primaire et ils ne peuvent donc être modifiés que par l'utilisateur. Si l'utilisateur veut que le fichier appartienne aux autres membres d'un groupe, il devra utiliser explicitement la commande **`chgrp`** pour changer l'appartenance.

Un autre avantage de cette approche est qu'elle unifie la gestion de la participation au groupe dans un seul fichier (**`/etc/group`**), au lieu d'un mélange entre **`/etc/passwd`** et **`/etc/group`**.

Pour chaque utilisateur, créez un groupe du même nom. Gérez la propriété du groupe avec **`/etc/group`** uniquement.

passwd_duplicate_username

Utilisation

```
<item>
 name: "passwd_duplicate_username"
 description: "This check makes sure that every username in /etc/passwd is unique."
</item>
```

Cette fonction intégrée assure que chaque nom d'utilisateur dans **`/etc/passwd`** est unique. Il réussit si tel est le cas et il échoue dans le cas contraire.

Les noms d'utilisateur dupliqués dans **`/etc/passwd`** causent des problèmes car il est difficile de savoir quels privilèges de compte sont utilisés.

La commande **`adduser`** interdit de créer un nom d'utilisateur dupliqué. Une telle configuration signifie en général que le système a été compromis, que les outils servant à assurer la gestion des utilisateurs comportent des bugs ou que le fichier **`/etc/passwd`** a été édité manuellement.

Supprimez les noms d'utilisateur dupliqués ou modifiez-les pour qu'ils soient différents.

passwd_duplicate_home

Utilisation

```
<item>
 name: "passwd_duplicate_home"
 description: "(arbitrary user comment)"
</item>
```

Cette fonction intégrée assure que tout utilisateur autre que les utilisateurs système (avec un UID supérieure à 100) dans **/etc/passwd** a un répertoire d'accueil unique.

Chaque nom d'utilisateur dans **/etc/passwd** doit avoir un répertoire d'accueil unique. Si des utilisateurs partagent le même répertoire d'accueil, l'un d'eux peut forcer l'autre à exécuter des commandes arbitraires en modifiant les fichiers de démarrage (**.profile**, etc.) ou en plaçant des binaires malveillants dans le répertoire d'accueil lui-même. En outre, un répertoire d'accueil partagé neutralise la responsabilité de l'utilisateur.

Les exigences de conformité spécifient que chaque utilisateur doit avoir un répertoire d'accueil unique.

passwd_shadowed

Utilisation

```
<item>
 name: "passwd_shadowed"
 description: "(arbitrary user comment)"
</item>
```

Ce contrôle intégré assure que chaque mot de passe dans **/etc/passwd** est « shadowed » (couvert), c'est-à-dire qu'il réside dans un autre fichier.

Puisque **/etc/passwd** est lisible universellement, le stockage des empreintes numériques de mot de passe des utilisateurs dans ce fichier permet à les utilisateurs qui peuvent y accéder d'y exécuter des programmes de craquage de mot de passe. Les tentatives pour deviner le mot de passe d'un utilisateur par une attaque en force (tentatives de connexion répétées en essayant des mots de passe différents à chaque fois) sont en général décelées dans les fichiers journaux du système. Si le fichier **/etc/passwd** contient les empreintes numériques de mot de passe, il pourrait être copié hors ligne et utilisé comme entrée d'un programme de craquage de mot de passe. Ceci donne à un attaquant la capacité d'obtenir les mots de passe d'utilisateur sans détection.

La plupart des systèmes Unix modernes ont des fichiers de mot de passe couverts. Consultez la documentation du système pour apprendre comment activer les mots de passe couverts sur le système.

passwd_invalid_gid

Utilisation

```
<item>
 name: "passwd_invalid_gid"
 description: "This check makes sure that every GID defined in /etc/passwd exists in
/etc/group."
</item>
```

Cette fonction intégrée assure que chaque ID de groupe (GID) répertorié dans **/etc/passwd** existe dans **/etc/group**. Elle réussit si chaque GID est correctement défini et échoue dans le cas contraire.

Chaque fois qu'un ID de groupe est défini dans **/etc/passwd**, il doit être immédiatement répertorié dans **/etc/group**. Sinon, le système est dans un état incohérent et des problèmes peuvent survenir.

Considérez le scénario suivant : un utilisateur (« bob ») a un UID de 1000 et un GID de 4000. Le GID n'est pas défini dans **/etc/group**, ce qui veut dire que le groupe primaire de l'utilisateur ne lui accordera aucun privilège aujourd'hui. Quelques mois plus tard, l'administrateur du système édite **/etc/group**, ajoute le groupe « admin » et sélectionne la GID « inutilisée » n°4000 pour l'identifier. L'utilisateur « bob » appartient à présent par défaut au groupe « admin », alors que ce n'était pas prévu.

Éditez **/etc/group** pour ajouter les GID manquants.

Gestion des fichiers de groupe

group_file_consistency

Utilisation

```
<item>
 name: "group_file_consistency"
 description: "This check makes sure /etc/group is valid."
</item>
```

Cette fonction intégrée assure que chaque ligne dans **/etc/group** a un format valide (par exemple trois éléments séparés par deux-points et une liste d'utilisateurs). Si une ligne est mal formée, elle est signalée et le contrôle échoue.

La présence d'un fichier **/etc/group** mal formé peut rendre inutilisables plusieurs outils de gestion des utilisateurs. Elle peut aussi indiquer une intrusion ou un bug dans une application personnalisée de gestion des utilisateurs. Elle peut également montrer que quelqu'un a essayé d'ajouter un utilisateur avec un nom de groupe non valide.

Éditez **/etc/group file to** pour corriger les lignes mal formées.

group_zero_gid

Utilisation

```
<item>
 name: "group zero gid"
 description: "This check makes sure that only ONE group has a gid of 0."
</item>
```

Cette fonction intégrée assure qu'un seul groupe a un ID de groupe (GID) de 0. Elle réussit si un groupe seulement a un GID de 0 et échoue dans le cas contraire.

Un GID de « 0 » signifie que les utilisateurs qui sont membres de ce groupe sont aussi membres du groupe primaire root. Ceci leur accorde des privilèges root sur tout fichier avec des permissions de groupe root.

Pour définir un groupe d'administrateurs, créez plutôt un groupe « admin ».

group_duplicate_name

Utilisation

```
<item>
 name: "group_duplicate_name"
 description: "This check makes sure that every group name in /etc/group is unique."
</item>
```

Ce contrôle intégré assure que chaque nom de groupe est unique. Il réussit si tel est le cas et il échoue dans le cas contraire.

Les noms d'utilisateur dupliqués dans **`/etc/group`** causent des problèmes car il est difficile de savoir quels privilèges de groupe sont utilisés. Cela veut dire qu'un nom de groupe dupliqué peut finir par avoir des membres ou des privilèges qu'il ne devrait pas avoir.

Supprimez ou renommez les noms de groupe dupliqués.

group_duplicate_gid

Utilisation

```
<item>
 name: "group_duplicate_gid"
 description: "(arbitrary user comment)"
</item>
```

Sur un système Unix, chaque groupe est identifié par son GID (group ID - ID de groupe), un nombre compris entre 0 et 65535. Si deux groupes ont le même GID, non seulement ils reçoivent les mêmes privilèges, mais le système les considère comme un même groupe. Ceci va à l'encontre de l'objectif visant à utiliser les groupes pour séparer les privilèges d'utilisateur.

Les normes de sécurité interdisent le partage d'un GID entre groupes. Si deux groupes doivent avoir les mêmes privilèges, ils devraient avoir les mêmes utilisateurs.

Supprimez les groupes dupliqués ou attribuez à l'un des groupes dupliqués un nouveau GID unique.

group_duplicate_members

Utilisation

```
<item>
 name: "group_duplicate_members"
 description: "This check makes sure that every member of a group is listed once."
</item>
```

Cette fonction intégrée assure que chaque membre d'un groupe est répertorié une seule fois. Elle réussit si chaque membre est unique et elle échoue dans le cas contraire.

Chaque membre du groupe devrait être répertorié une seule fois. Bien que le fait de figurer plusieurs fois dans la liste ne cause pas de problème au niveau du système d'exploitation sous-jacent, cela complique la tâche de l'administrateur système car la révocation des privilèges est plus complexe. Par exemple, si le groupe « admin » a les membres « alice,bob,charles,daniel,bob », « bob » devra être retiré deux fois si ses privilèges sont révoqués.

Vérifiez que chaque membre est répertorié une seule fois.

group_nonexistant_users

Utilisation

`<item>`
` name: "group_nonexistant_users"`
` description: "This check makes sure that every member of a group actually exists."`
`</item>`

Ce contrôle assure que chaque membre d'un groupe existe réellement dans **/etc/passwd**.

Les utilisateurs non existants dans **/etc/group** indiquent des pratiques administratives incomplètes. L'utilisateur n'existe pas à cause d'une erreur typographique ou parce qu'il n'a pas été retiré du groupe lorsqu'il a été retiré du système.

Il n'est pas recommandé que des utilisateurs « fantômes » restent dans **/etc/group**. Si un utilisateur possédant le même nom d'utilisateur était ajouté à une date ultérieure, cet utilisateur pourrait avoir des privilèges de groupe qui ne devraient normalement pas lui être accordés.

Retirez les utilisateurs non existants de **/etc/group**.

Environnement racine

dot_in_root_path_variable

Utilisation

`<item>`
` name: "dot_in_root_path_variable"`
` description: "This check makes sure that root's $PATH variable does not contain any relative path."`
`</item>`

Ce contrôle assure que le répertoire de travail actuel (« . ») n'est pas inclus dans le chemin exécutable de l'utilisateur root. Ceci empêche qu'un utilisateur malveillant augmente ses privilèges au niveau superutilisateur en forçant un administrateur en session root à exécuter un cheval de Troie qui peut être installé dans le répertoire de travail actuel.

writeable_dirs_in_root_path_variable

Utilisation

`<item>`
` name: "writeable_dirs_in_root_path_variable"`
` description: "This check makes sure that root's $PATH variable does not contain any writeable directory."`
`</item>`

Ce contrôle signale tous les répertoires inscriptibles universellement/par le groupe dans la variable PATH des utilisateurs root. Tous les répertoires renvoyés par ce contrôle doivent être examinés avec soin et les permissions non nécessaires inscriptibles universellement/par le groupe sur les répertoires doivent être retirées comme suit :

```
# chmod go-w path/to/directory
```

Permissions de fichier

find_orphan_files

Utilisation

```
<item>
 name: "find_orphan_files"
 description: "This check finds all the files which are 'orphaned' (ie: whose owner is
an invalid UID or GID)."
 # Globs allowed (? and *)
 (optional) basedir: "<directory>"
 (optional) ignore: "<directory>"
 (optional) dir: "<directory>"
</item>
```

Ce contrôle signale tous les fichiers qui n'appartiennent à aucun utilisateur sur le système.

Par défaut, la recherche est effectuée de façon récurrente sous le répertoire « / ». Ceci peut ralentir considérablement l'exécution du contrôle suivant le nombre de fichiers présents sur le système distant. Toutefois, le répertoire de base par défaut pour la recherche peut être changé, si nécessaire, en utilisant le mot clé optionnel **basedir**. Il est aussi possible d'omettre la recherche de certains fichiers dans un répertoire de base en utilisant un autre mot clé optionnel **ignore**. Lors de la recherche de systèmes de fichier, le contrôle ignore par défaut tout répertoire monté sur NFS, sauf s'il a été spécifié avec le mot clé optionnel **dir**.

À cause de la nature du contrôle, il est normal que son exécution prenne plusieurs heures, suivant le type de système scanné. Une valeur de temporisation par défaut, à savoir le délai à l'issue duquel Nessus arrêtera de traiter les résultats de ce contrôle, a été paramétrée à cinq heures et cette valeur ne peut pas être changée.

Exemple :

```
 <item>
  name: "find_orphan_files"
  description: "This check finds all the files which are 'orphaned' (ie: whose owner is
        an invalid UID or GID)."
  # Globs allowed (? and *)
  basedir: "/tmp"
  ignore: "/tmp/foo"
  ignore: "/tmp/b*"
 </item>
```

find_world_writeable_files

Utilisation

```
<item>
 name: "find_world_writeable_files"
 description: "This check finds all the files which are world writeable and whose sticky
bit is not set."
 # Globs allowed (? and *)
 (optional) basedir: "<directory>"
 (optional) ignore: "<directory>"
 (optional) dir: "<directory>"
```

```
</item>
```

Ce contrôle signale tous les fichiers qui sont inscriptibles universellement sur le système distant. Idéalement, il ne devrait pas exister de fichier inscriptible universellement sur le système distant, et le résultat de ce contrôle ne devrait rien indiquer. Toutefois, les besoins de l'organisation peuvent exiger la présence de fichiers inscriptibles universellement. Tous les éléments renvoyés par ce contrôle doivent être vérifiés avec soin et les fichiers qui n'ont pas nécessairement besoin d'attributs inscriptibles universellement doivent être retirés comme suit :

```
  # chmod o-w world_writeable_file
```

Par défaut, la recherche est effectuée de façon récurrente sous le répertoire « / ». Ceci peut ralentir considérablement l'exécution du contrôle suivant le nombre de fichiers présents sur le système distant. Toutefois, le répertoire de base par défaut pour la recherche peut être changé, si nécessaire, en utilisant le mot clé optionnel **basedir**. Il est aussi possible d'omettre la recherche de certains fichiers dans un répertoire de base en utilisant un autre mot clé optionnel **ignore**. Lors de la recherche de systèmes de fichier, le contrôle ignore par défaut tout répertoire monté sur NFS, sauf s'il a été spécifié avec le mot clé optionnel **dir**.

À cause de la nature du contrôle, il est normal que son exécution prenne plusieurs heures, suivant le type de système scanné. Une valeur de temporisation par défaut, à savoir le délai à l'issue duquel Nessus arrêtera de traiter les résultats de ce contrôle, a été paramétrée à cinq heures et cette valeur ne peut pas être changée.

Exemple :

```
  <item>
   name: "find_world_writeable_files"
   description: "Search for world-writable files"
   # Globs allowed (? and *)
   basedir: "/tmp"
   ignore: "/tmp/foo"
   ignore: "/tmp/bar"
  </item>
```

find_world_writeable_directories

Utilisation

```
<item>
 name: "find_world_writeable_directories"
 description: "This check finds all the directories which are world writeable and whose
sticky bit is not set."
 # Globs allowed (? and *)
 (optional) basedir: "<directory>"
 (optional) ignore: "<directory>"
 (optional) dir: "<directory>"
</item>
```

Ce contrôle signale tous les répertoires qui sont inscriptibles universellement et dont le bit de rappel n'est pas défini sur le système distant. La vérification de la définition du bit de rappel pour tous les répertoires inscriptibles universellement assure que seul le propriétaire d'un fichier dans un répertoire peut supprimer le fichier. Ceci empêche tout autre utilisateur de supprimer le fichier accidentellement ou intentionnellement.

Par défaut, la recherche est effectuée de façon récurrente sous le répertoire « / ». Ceci peut ralentir considérablement l'exécution du contrôle suivant le nombre de fichiers présents sur le système distant. Toutefois, le répertoire de base par défaut pour la recherche peut être changé, si nécessaire, en utilisant le mot clé optionnel **basedir**. Il est aussi possible d'omettre la recherche de certains fichiers dans un répertoire de base en utilisant un autre mot clé optionnel **ignore**. Lors de la recherche de systèmes de fichier, le contrôle ignore par défaut tout répertoire monté sur NFS, sauf s'il a été spécifié avec le mot clé optionnel **dir**.

À cause de la nature du contrôle, il est normal que son exécution prenne plusieurs heures, suivant le type de système scanné. Une valeur de temporisation par défaut, à savoir le délai à l'issue duquel Nessus arrêtera de traiter les résultats de ce contrôle, a été paramétrée à cinq heures et cette valeur ne peut pas être changée.

Exemple :

```
 <item>
  name: "find_world_writeable_directories"
  description: "This check finds all the directories which are world writeable and
        whose sticky bit is not set."
  # Globs allowed (? and *)
  basedir: "/tmp"
  ignore: "/tmp/foo"
  ignore: "/tmp/b*"
 </item>
```

find_world_readable_files

Utilisation

```
<item>
 name: "find_world_readable_files"
 description: "This check finds all the files in a directory with world readable
permissions."
 # Globs allowed (? and *)
 (optional) basedir: "<directory>"
 (optional) ignore: "<directory>"
 (optional) dir: "<directory>"
</item>
```

Ce contrôle vérifie que tous les fichiers sont lisibles universellement. La recherche de fichiers lisibles, par exemple dans les répertoires d'accueil de l'utilisateur, garantit qu'aucun autre utilisateur (par exemple, clés SSH privées) ne peut accéder aux fichiers sensibles.

Par défaut, la recherche est effectuée de façon récurrente sous le répertoire « / ». Ceci peut ralentir considérablement l'exécution du contrôle suivant le nombre de fichiers présents sur le système distant. Toutefois, le répertoire de base par défaut pour la recherche peut être changé, si nécessaire, en utilisant le mot clé optionnel **basedir**. Il est aussi possible d'omettre la recherche de certains fichiers dans un répertoire de base en utilisant un autre mot clé optionnel **ignore**. Lors de la recherche de systèmes de fichier, le contrôle ignore par défaut tout répertoire monté sur NFS, sauf s'il a été spécifié avec le mot clé optionnel **dir**.

À cause de la nature du contrôle, il est normal que son exécution prenne plusieurs heures, suivant le type de système scanné. Une valeur de temporisation par défaut, à savoir le délai à l'issue duquel Nessus arrêtera de traiter les résultats de ce contrôle, a été paramétrée à cinq heures et cette valeur ne peut pas être changée.

Exemple :

```
<item>
 name: "find_world_readable_files"
 description: "This check finds all the files in a directory with world readable
       permissions."
 basedir: "/home"
 ignore: "/home/tmp"
 dir: "/home/extended"
</item>
```

find_suid_sgid_files

Utilisation

```
<item>
 name: "find_suid_sgid_files"
 description: "This check finds all the files which have their SUID or SGID bit set."
 # Globs allowed (? and *)
 (optional) basedir: "<directory>"
 (optional) ignore: "<directory>"
 (optional) dir: "<directory>"
</item>
```

Ce contrôle signale tous les fichiers avec le jeu de bits SUID/SGID. Tous les fichiers signalés par ce contrôle doivent être vérifiés avec soin, en particulier les scripts shell et les exécutables internes, par exemple les exécutables qui n'ont pas été expédiés avec le système. Les fichiers SUID/SGID présentent le risque d'accroître les privilèges d'un utilisateur normal à ceux détenus par le propriétaire ou le groupe du fichier. Lorsque de tels fichiers/scripts doivent exister, ils devraient être examinés avec soin pour vérifier s'ils permettent la création d'un fichier avec des privilèges élevés.

Par défaut, la recherche est effectuée de façon récurrente sous le répertoire « / ». Ceci peut ralentir considérablement l'exécution du contrôle suivant le nombre de fichiers présents sur le système distant. Toutefois, le répertoire de base par défaut pour la recherche peut être changé, si nécessaire, en utilisant le mot clé optionnel **basedir**. Il est aussi possible d'omettre la recherche de certains fichiers dans un répertoire de base en utilisant un autre mot clé optionnel **ignore**. Lors de la recherche de systèmes de fichier, le contrôle ignore par défaut tout répertoire monté sur NFS, sauf s'il a été spécifié avec le mot clé optionnel **dir**.

À cause de la nature du contrôle, il est normal que son exécution prenne plusieurs heures, suivant le type de système scanné. Une valeur de temporisation par défaut, à savoir le délai à l'issue duquel Nessus arrêtera de traiter les résultats de ce contrôle, a été paramétrée à cinq heures et cette valeur ne peut pas être changée.

Exemple :

```
<item>
 name: "find suid sgid files"
 description: "Search for SUID/SGID files"
 # Globs allowed (? and *)
 basedir: "/"
 ignore: "/usr/sbin/ping"
</item>
```

home_dir_localization_files_user_check
Ce contrôle intégré vérifie si un fichier de localisation placé dans le répertoire d'accueil d'un utilisateur est la propriété de l'utilisateur ou de la racine (root).

Un ou plusieurs fichiers peuvent être répertoriés à l'aide du jeton « file ». Cependant, si le jeton « file » est manquant, le contrôle recherche par défaut les fichiers suivants :

- **.login**
- **.cschrc**
- **.logout**
- **.profile**
- **.bash_profile**
- **.bashrc**
- **.bash_logout**
- **.env**
- **.dtprofile**
- **.dispatch**
- **.emacs**
- **.exrc**

Exemple :

```
<item>
 name: "home_dir_localization_files_user_check"
 description: "Check file .foo/.foo2"
 file: ".foo"
 file: ".foo2"
 file: ".foo3"
</item>
```

home_dir_localization_files_group_check
Ce contrôle intégré vérifie si un fichier de localisation placé dans le répertoire d'accueil d'un utilisateur est la propriété d'un groupe : le groupe principal de l'utilisateur ou la racine (root).

Un ou plusieurs fichiers peuvent être répertoriés à l'aide du jeton « file ». Cependant, si le jeton « file » est manquant, le contrôle recherche par défaut les fichiers suivants :

- **.login**
- **.cschrc**
- **.logout**
- **.profile**
- **.bash_profile**
- **.bashrc**
- **.bash_logout**
- **.env**
- **.dtprofile**
- **.dispatch**
- **.emacs**
- **.exrc**

Exemple :

```
<item>
 name: "home_dir_localization_files_group_check"
 description: "Check file .foo/.foo2"
 file: ".foo"
```

```
  file: ".foo2"
  file: ".foo3"
 </item>
```

Contenu de fichier suspect
admin_accounts_in_ftpusers

Utilisation

```
<item>
 name: "admin_accounts_in_ftpusers"
 description: "This check makes sure every account whose UID is below 500 is present in
/etc/ftpusers."
</item>
```

Ce contrôle vérifie si tous les comptes admin (les utilisateurs avec un UID inférieur à 500) sont présents dans **/etc/ftpusers, /etc/ftpd/ftpusers ou /etc/vsftpd.ftpusers**.

Fichiers non nécessaires
find_pre-CIS_files

Utilisation

```
<item>
 name: "find_preCIS_files"
 description: "Find and list all files created by CIS backup script."
 # Globs allowed (? and *)
 (optional) basedir: "<directory>"
 (optional) ignore: "<directory>"
</item>
```

Ce contrôle est personnalisé pour une exigence CIS (Center for Internet Security, Centre pour la sécurité Internet) spécifique concernant la certification pour la norme Red Hat CIS. Ce contrôle est particulièrement utile pour un utilisateur qui pourrait avoir configuré/renforcé un système Red Hat à partir de la norme Red Hat de CIS. L'outil de la norme CIS fournit un script de secours pour sauvegarder tous les fichiers système qui peuvent être modifiés pendant le processus de renforcement du système et ces fichiers sont dotés d'un suffixe avec un mot clé « **-preCIS** ». Ces fichiers devraient être supprimés une fois que toutes les recommandations de la norme ont été effectuées avec succès et que le système a été restauré à son état fonctionnel. Ce contrôle assure qu'aucun fichier « **preCIS** » n'existe sur le système distant.

Par défaut, la recherche est effectuée de façon récurrente sous le répertoire « / ». Ceci peut ralentir considérablement l'exécution du contrôle suivant le nombre de fichiers présents sur le système distant. Toutefois, le répertoire de base par défaut pour la recherche peut être changé, si nécessaire, en utilisant le mot clé optionnel **basedir**. Il est aussi possible d'omettre la recherche de certains fichiers dans un répertoire de base en utilisant un autre mot clé optionnel **ignore**.

À cause de la nature du contrôle, il est normal que son exécution prenne plusieurs heures, suivant le type de système scanné. Une valeur de temporisation par défaut, à savoir le délai à l'issue duquel Nessus arrêtera de traiter les résultats de ce contrôle, a été paramétrée à cinq heures et cette valeur ne peut pas être changée.

Conditions
Il est possible de définir la logique **if/then/else** dans la stratégie Unix. Ceci permet à l'utilisateur final d'utiliser un seul fichier capable de traiter plusieurs configurations. Par exemple, le même fichier de stratégie peut vérifier les paramètres pour Postfix et Sendmail en utilisant la syntaxe correcte **if/then/else**.

La syntaxe pour exécuter les conditions est la suivante :

```
<if>
 <condition type: "or">
  <Insert your audit here>
 </condition>
 <then>
  <Insert your audit here>
 </then>
 <else>
  <Insert your audit here>
 </else>
</if>
```

Exemple :

```
<if>
 <condition type: "or">
  <custom_item>
   type: FILE_CHECK
   description: "Make sure /etc/passwd contains root"
   file: "/etc/passwd"
   owner: "root"
  </custom_item>
 </condition>

 <then>
  <custom_item>
   type: FILE_CONTENT_CHECK
   description: "Make sure /etc/passwd contains root (then)"
   file: "/etc/passwd"
   regex: "^root"
   expect: "^root"
  </custom_item>
 </then>

 <else>
  <custom_item>
   type: FILE_CONTENT_CHECK
   description: "Make sure /etc/passwd contains root (else)"
   file: "/etc/passwd"
   regex: "^root"
   expect: "^root"
  </custom_item>
 </else>
</if>
```

L'échec ou la réussite de la condition n'est jamais indiqué dans le rapport car il s'agit d'une vérification « muette ».

Les conditions peuvent être du type « **and** » ou « **or** ».

NetApp Data ONTAP

Cette section décrit le format et les fonctions des systèmes de stockage exécutant les contrôles de conformité NetApp Data ONTAP, ainsi que les raisons de chaque paramètre.

Emploi des guillemets :

Les guillemets simples et doubles sont interchangeables autour des champs d'audit, sauf dans les deux cas suivants :

1. Dans les contrôles de conformité Windows où des champs spéciaux tels que CRLF, doivent être interprétés littéralement, utilisez des guillemets simples. Tout champ intégré qui doit être interprété comme une chaîne doit être inclus dans une séquence d'échappement.

Par exemple :
```
expect: 'First line\r\nSecond line\r\nJohn\'s Line'
```

2. Des guillemets doubles sont requis lorsque vous utilisez les fichiers Windows « include_paths » et « exclude_paths ».

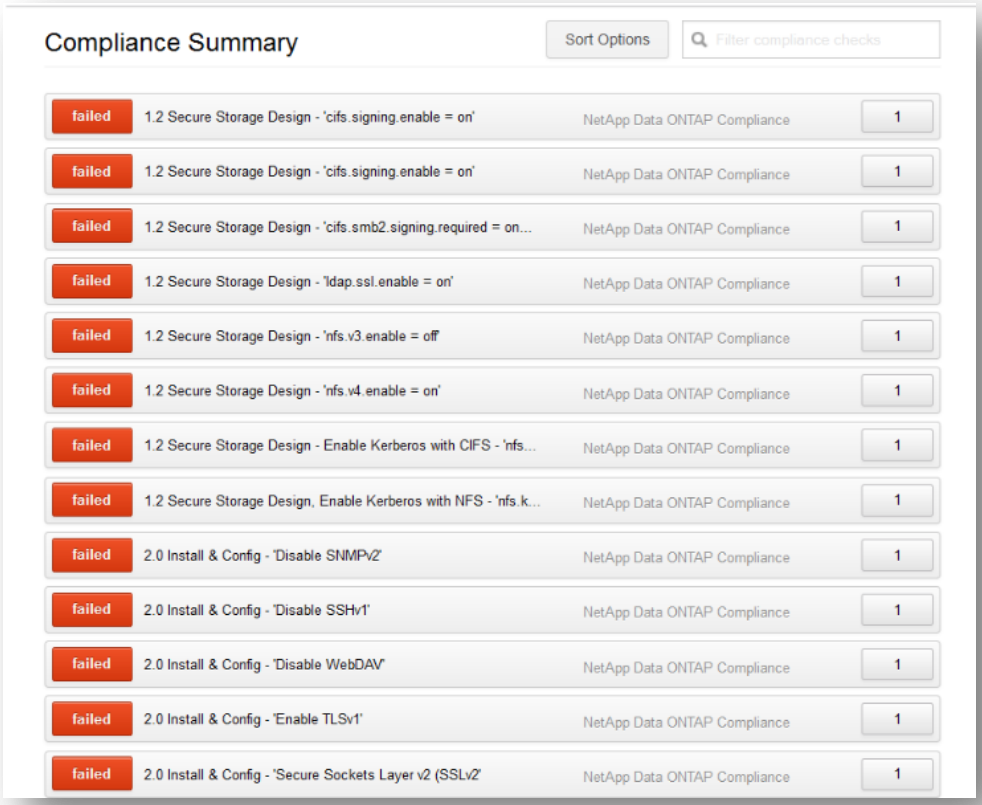
Si des chaînes sont utilisées dans tout type de champ (description, value_data, regex, etc.) contenant des guillemets simples ou doubles, il y a deux façons de les traiter :

a. Utilisez le type de guillemets opposés pour les guillemets extérieurs.

Par exemple :
```
expect: "This is John's Line"
expect: 'We are looking for a double-quote-".*'
```

b. Échappez tout guillemet intégré avec une barre oblique inverse (guillemets doubles uniquement).

Par exemple :
```
expect: "\"Text to be searched\""
```



Privilèges utilisateur requis

Pour effectuer avec succès un scan de conformité sur un système NetApp Data ONTAP, les utilisateurs authentifiés doivent posséder les privilèges définis ci-dessous :

Identifiants `root` pour filtre NetApp Data ONTAP

En plus des privilèges ci-dessus, une stratégie d'audit pour les contrôles de conformité NetApp Data ONTAP et le plugin Nessus avec l'ID 66934 (Contrôles de conformité NetApp Data ONTAP) sont requis.

Pour exécuter un scan sur le périphérique, commencez par créer la stratégie d'audit. Utilisez ensuite le menu « **SSH settings** » (Paramètres SSH) sous l'onglet « **Credentials** » (Identifiants) de la stratégie pour fournir les identifiants `root`. Sous l'onglet « **Plugins** » (Plugins) de la stratégie, sélectionnez la famille de plugins « Policy Compliance » (Conformité des stratégies) et activez le plugin avec l'ID 66934 nommé « **NetApp Data ONTAP Compliance Checks** » (Contrôles de conformité NetApp Data ONTAP). Sous l'onglet « **Preferences** » (Préférences), sélectionnez alors le menu déroulant « NetApp Data ONTAP Compliance Checks » (Contrôles de conformité NetApp Data ONTAP) et ajoutez le fichier **`.audit`** NetApp à partir du portail d'assistance de Tenable. Enfin, enregistrez la stratégie et exécutez le scan.

S'il n'est pas possible de fournir les identifiants `root`, un compte reposant sur des privilèges moindres peut être créé pour faciliter l'audit :

1. Créez un nouveau rôle (par exemple, `nessus_audit`):

   ```
   # role add nessus_audit -a login-ssh,cli-version,cli-options,cli-uptime
   ```

2. Affectez le rôle à un groupe (par exemple, `nessus_admins`) :

   ```
   # group add nessus_admins -r nessus_audit
   ```

3. Affectez le groupe à un utilisateur :

```
# useradmin user add nessus -g nessus_admins
```

Check Type: CONFIG_CHECK

Les contrôles de conformité Check Point sont placés entre crochets dans l'encapsulation **custom_item** et CONFIG_CHECK. Ils sont traités comme les autres fichiers **.audit** et ils s'appliquent aux systèmes qui utilisent le système d'exploitation NetApp Data ONTAP. La vérification « CONFIG_CHECK » comprend au moins deux mots clés. Les mots clés **type** et **description** sont obligatoires et ils sont suivis d'un ou de plusieurs mots clés. Le contrôle vérifie la sortie de la commande « options ».

Mots clés

Le tableau suivant indique comment chaque mot clé peut être utilisé dans les contrôles de conformité NetApp Data ONTAP :

Mot clé	Exemple d'utilisation et paramètres pris en charge
type	« CHECK_CONFIG » détermine si l'élément de config spécifié existe dans la sortie « show configuration » de NetApp Data ONTAP.
description	Le mot clé « **description** » permet d'ajouter une brève description du contrôle en cours. Il est fortement recommandé que le champ **description** soit unique et qu'aucun autre contrôle ne possède le même champ **description**. SecurityCenter de Tenable utilise ce champ pour produire automatiquement un numéro d'ID de plugin unique basé sur le champ **description**. Exemple : `description: "1.0 Require strong Password Controls - 'min-password-length >= 8'"`
info	Le mot clé « **info** » est utilisé pour ajouter une description plus détaillée au contrôle en cours. La raison de ce contrôle pourrait notamment être une réglementation, une URL avec plus d'informations ou une stratégie d'entreprise. Plusieurs champs **info** peuvent être ajoutés sur des lignes séparées pour formater le texte sous forme de paragraphe. Il n'existe pas de limite prédéfinie pour le nombre de champs **info** qui peuvent être utilisés.  Chaque balise « **info** » doit être écrite sur une ligne séparée sans saut de ligne. Si plusieurs lignes sont requises (par exemple, pour des raisons de formatage), ajoutez des balises « **info** » supplémentaires. Exemple : `info: "Enable palindrome-check on passwords"`
severity	Le mot clé « **severity** » spécifie la gravité du contrôle en cours. Exemple : `severity: MEDIUM` La gravité peut être paramétrée sur HIGH (Élevée), MEDIUM (Moyenne) ou LOW (Faible).
regex	Le mot clé « **regex** » permet de chercher le paramètre d'élément de configuration pour trouver une expression rationnelle particulière.

	Exemple : `regex: "set snmp .+"` Les caractères génériques suivants nécessitent un traitement spécial : + \ * ( ) ^ Échapper deux fois ces caractères avec deux barres obliques inverses « \\ » ou les placer entre crochets « [] » s'ils doivent être interprétés littéralement. D'autres caractères tels que ceux indiqués ci-dessous nécessitent seulement une barre oblique inverse pour être interprétés littéralement : . ? " ' La raison provient de la façon dont le compilateur traite ces caractères. Si une balise « regex » est définie pour un contrôle, mais pas une balise « expect », « not_expect » ou « number_of_lines », le contrôle signale simplement toutes les lignes correspondant au mot clé regex.

expect	Ce mot clé permet de vérifier l'élément de configuration qui correspond à la balise « **regex** ».Si la balise « **regex** » n'est pas utilisée, il recherche la chaîne « **expect** » dans l'ensemble de config.
	
	Le contrôle renvoie un résultat concluant lorsque la ligne de config trouvée par « **regex** » correspond à la balise « **expect** » ou, si « **regex** » n'est pas défini, lorsque la chaîne « **expect** » est trouvée dans config.
	
	Exemple :
	`regex: "set password-controls complexity"`
	`expect: "set password-controls complexity [1-4]"`
	
	Dans le cas ci-dessus, la balise « **expect** » garantit que la complexité est définie avec une valeur comprise entre 1 et 4.

not_expect	Ce mot clé permet de rechercher les éléments de configuration qui ne devraient pas figurer dans la configuration.
	
	Il a l'action contraire de « **expect** ». Le contrôle est concluant lorsque la ligne de config trouvée par « **regex** » ne correspond pas à la balise « **not_expect** » ou, si « **regex** » n'est pas défini, lorsque la chaîne « **not_expect** » n'est pas trouvée dans config.
	
	Exemple :
	`regex: "set password-controls password-expiration"`
	`not_expect: "set password-controls password-expiration never"`
	
	Dans le cas ci-dessus, la balise « **not_expect** » garantit que password-controls n'est pas paramétré sur « never » (jamais).

Exemples de CONFIG_CHECK

Les exemples qui suivent illustrent l'utilisation de CONFIG_CHECK pour un périphérique NetApp Data ONTAP :

```
<custom_item>
 type: CONFIG_CHECK
 description: "1.2 Secure Storage Design, Enable Kerberos with NFS -
       'nfs.kerberos.enable = on'"
 info: "NetApp recommends the use of security features in IP storage protocols to
       secure client access"
 solution: "Enable Kerberos with NFS"
```

```
 reference: "PCI|2.2.3"
 see_also: "http://media.netapp.com/documents/tr-3649.pdf"
 regex: "nfs.kerberos.enable[\\s\\t]+"
 expect: "nfs.kerberos.enable[\\s\\t]+on"
</custom_item>
```

Conditions

Il est possible de définir la logique **if/then/else** dans la stratégie d'audit NetApp Data ONTAP. Ceci permet à l'utilisateur final d'utiliser un seul fichier capable de traiter plusieurs configurations.

La syntaxe pour exécuter les conditions est la suivante :

```
<if>
  <condition type:"or">
      < Insert your audit here >
  </condition>
 <then>
      < Insert your audit here >
 </then>
 <else>
      < Insert your audit here >
 </else>
</if>
```

Exemple :

```
<if>
 <condition type: "OR">
 <custom_item>
  type: CONFIG_CHECK
  description: "2.6 Install and configure Encrypted Connections to devices - 'telnet'"
  regex: "set net-access telnet"
  expect: "set net-access telnet off"
  info: "Do not use plain-text protocols."
</custom_item>
 </condition>
 <then>
  <report type: "PASSED">
   description: "Telnet is disabled"
  </report>
 </then>
 <else>
  <custom_item>
  type: CONFIG_CHECK
  description: "2.6 Install and configure Encrypted Connections to devices - 'telnet'"
  regex: "set net-access telnet"
  expect: "set net-access telnet off"
  info: "Do not use plain-text protocols."
</custom_item>
 </else>
</if>
```

La condition n'est jamais indiquée dans le rapport, qu'elle ait échoué ou non, car il s'agit d'une vérification « muette ».

Les conditions peuvent être du type « **and** » ou « **or** ».

Rapports

Peut être exécuté dans une instruction <then> ou <else> pour parvenir à la condition PASSED/FAILED voulue.

```
<if>
 <condition type: "OR">
 <custom_item>
  type: CONFIG_CHECK
  description: "2.6 Install and configure Encrypted Connections to devices - 'telnet'"
  regex: "set net-access telnet"
  expect: "set net-access telnet off"
  info: "Do not use plain-text protocols."
</custom_item>
 </condition>
 <then>
  <report type: "PASSED">
   description: "Telnet is disabled"
  </report>
 </then>
 <else>
 <report type: "FAILED">
   description: "Telnet is disabled"
  </report>
 </else>
</if>
```

PASSED (Réussite), WARNING (Avertissement) et FAILED (Échec) sont des valeurs acceptables pour « report type ».

Référence des fichiers de conformité d'audit pour la configuration IBM iSeries

Cette section décrit le format et les fonctions de contrôles de conformité IBM iSeries, ainsi que les raisons de chaque paramètre.

Emploi des guillemets :

Les guillemets simples et doubles sont interchangeables autour des champs d'audit, sauf dans les deux cas suivants :

1. Dans les contrôles de conformité Windows où des champs spéciaux tels que CRLF, doivent être interprétés littéralement, utilisez des guillemets simples. Tout champ intégré qui doit être interprété comme une chaîne doit être inclus dans une séquence d'échappement.

Par exemple :
`expect: 'First line\r\nSecond line\r\nJohn\'s Line'`

2. Des guillemets doubles sont requis lorsque vous utilisez les fichiers Windows « include_paths » et « exclude_paths ».

Si des chaînes sont utilisées dans tout type de champ (description, value_data, regex, etc.) contenant des guillemets simples ou doubles, il y a deux façons de les traiter :

a. Utilisez le type de guillemets opposés pour les guillemets extérieurs.

Par exemple :

```
expect: "This is John's Line"
expect: 'We are looking for a double-quote-".*'
```

b. Échappez tout guillemet intégré avec une barre oblique inverse (guillemets doubles uniquement).

Par exemple :

```
expect: "\"Text to be searched\""
```

Privilèges utilisateur requis

Pour effectuer avec succès un scan de conformité sur un système iSeries, les utilisateurs authentifiés doivent posséder les privilèges définis ci-dessous :

1. Un utilisateur doté de l'autorité (*ALLOBJ) ou audit (*AUDIT) peut vérifier toutes les valeurs système. Un tel utilisateur appartient normalement à la classe (*SECOFR).

2. Les utilisateurs de la classe (*USER) ou (*SYSOPR) peuvent vérifier la plupart des valeurs, sauf QAUDCTL, QAUDENDACN, QAUDFRCLVL, QAUDLVL, QAUDLVL2 et QCRTOBJAUD.

Si un utilisateur ne dispose pas des privilèges lui permettant d'accéder à une valeur, la valeur renvoyée sera *NOTAVL.

Type de contrôle

Tous les contrôles de conformité IBM iSeries doivent être mis entre crochets avec l'encapsulation **`check_type`** et la désignation « AS/400 ». Ceci est requis pour distinguer les fichiers **`.audit`** spécifiquement conçus pour les systèmes exécutant un système IBM iSeries des autres types d'audits de conformité.

Exemple :

```
<check_type:"AS/400">
```

Contrairement à d'autres types d'audit de conformité, aucun mot clé supplémentaire de type ou de version n'est disponible.

Mots clés

Le tableau suivant indique comment chaque mot clé peut être utilisé dans les contrôles de conformité IBM iSeries :

Mot clé	Exemple d'utilisation et paramètres pris en charge
`type`	`AUDIT_SYSTEMVAL` `SHOW_SYSTEMVAL`
`systemvalue`	Ce mot clé permet de spécifier une valeur donnée à contrôler dans le système IBM iSeries. Exemple : `systemvalue: "QALWUSRDMN"`
`description`	Ce mot clé permet d'ajouter une brève description du contrôle en cours. Il est fortement recommandé que le champ **`description`** soit unique et qu'aucun autre contrôle ne possède le même champ. SecurityCenter de Tenable utilise ce champ pour produire automatiquement un numéro d'ID de plugin unique basé sur le champ **`description`**. Exemple : `description: "Allow User Domain Objects (QALWUSRDMN) - '*all'"`

value_type	Ce mot clé est utilisé pour définir le type de valeur (« POLICY_DWORD » ou « POLICY_TEXT ») en cours de contrôle sur le système IBM iSeries.
	
	Exemple :
	`value_type: "POLICY_DWORD"`
	
	Exemple :
	`value_type: "POLICY_TEXT"`

value_data	Ce mot clé définit une valeur de données attendue pour une valeur système. Exemple : `value_type: "^([6-9]|[1-9][0-9]+)$"`

check_type	Ce mot clé définit le type de contrôle utilisé par rapport à une valeur de donnée.
	
	Exemples :
	`check_type: "CHECK_EQUAL"`
	`check_type: "CHECK_NOT_EQUAL"`
	`check_type: "CHECK_GREATER_THAN"`
	`check_type: "CHECK_GREATER_THAN_OR_EQUAL"`
	`check_type: "CHECK_LESS_THAN"`
	`check_type: "CHECK_LESS_THAN_OR_EQUAL"`
	`check_type: "CHECK_REGEX"`
	
	Exemple :
	`<custom_item>`
	`type: AUDIT_SYSTEMVAL`
	`systemvalue: "QUSEADPAUT"`
	`description: "Use Adopted Authority (QUSEADPAUT) - '!= *none'"`
	`value_type: POLICY_TEXT`
	`value_data: "*none"`
	`check_type: CHECK_NOT_EQUAL`
	`</custom_item>`

info	Ce mot clé est utilisé pour ajouter une description plus détaillée du contrôle en cours comme une réglementation, une URL, une stratégie d'entreprise ou une autre raison pour laquelle le paramètre est requis. Plusieurs champs **info** peuvent être ajoutés sur des lignes séparées pour formater le texte sous forme de paragraphe. Il n'existe pas de limite prédéfinie pour le nombre de champs **info** qui peuvent être utilisés. Exemple : `info: "\nref : http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/books/sc415302.pdf pg. 21"`

Éléments personnalisés

Un élément personnalisé est un contrôle complet défini en fonction des mots clés ci-dessus. Une liste de types d'éléments personnalisés disponibles est indiquée ci-dessous. Chaque contrôle commence par une balise « **<custom_item>** » et se termine par « **</custom_item>** ». Les balises contiennent des listes d'un ou plusieurs mots clés qui sont interprétés par l'analyseur syntaxique de contrôle de conformité pour effectuer les contrôles.

Les contrôles d'audit personnalisé peuvent utiliser indifféremment « **</custom_item>** » et « **</item>** » pour la balise de fermeture.

AUDIT_SYSTEMVAL

« AUDIT_SYSTEMVALUE » vérifie la valeur du paramètre de configuration identifié par le mot clé « **systemvalue** ». Ce type de comparaison par rapport à la valeur en cours de vérification est spécifié par le mot clé « **check_type** ».

```
<custom_item>
 type: AUDIT_SYSTEMVAL
 systemvalue: "QALWUSRDMN"
 description: "Allow User Domain Objects (QALWUSRDMN) - '*all'"
 value_type: POLICY_TEXT
 value_data: "*all"
 info: "\nref :
       http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/books/sc415302.pdf
       pg. 21"
</custom_item>
```

SHOW_SYSTEMVAL

L'audit « SHOW_SYSTEMVAL » signale uniquement la valeur du paramètre de configuration identifié par le mot clé « **systemvalue** ».

```
<custom_item>
 type: SHOW_SYSTEMVAL
 systemvalue: "QAUDCTL"
 description: "show QAUDCTL value"
 severity: MEDIUM
</custom_item>
```

Conditions

Il est possible de définir la logique **if/then/else** dans la stratégie IBM iSeries. Ceci permet à l'utilisateur final de renvoyer un message d'avertissement plutôt qu'un message de réussite/échec en cas d'audit concluant.

La syntaxe pour exécuter les conditions est la suivante :

```
<if>
 <condition type: "or">
  <Insert your audit here>
 </condition>
 <then>
  <Insert your audit here>
 </then>
 <else>
  <Insert your audit here>
 </else>
</if>
```

Exemple :

```
<if>
 <condition type: "or">
  <custom_item>
   type: AUDIT_SYSTEMVAL
   systemvalue: "QDSPSGNINF"
   description: "Sign-on information is displayed (QDSPSGNINF)"
```

```
    info: "\nref :
        http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/books/sc415302.pdf
        pg. 23"
    value_type: POLICY_DWORD
    value_data: "1"
   </custom_item>
  </condition>

  <then>
   <custom_item>
    type: AUDIT_SYSTEMVAL
    systemvalue: "QDSPSGNINF"
    description: "Sign-on information is not displayed (QDSPSGNINF)"
    info: "\nref :
        http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/books/sc415302.pdf
        pg. 23"
    value_type: POLICY_DWORD
    value_data: "1"
   </custom_item>
  </then>

  <else>
   <report type: "WARNING">
    description: "Sign-on information is displayed (QDSPSGNINF)"
    info: ""\nref :
        http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/books/sc415302.pdf
        pg. 23"
    info: "Check system policy to confirm requirements."
   </report>
  </else>
 </if>
```

L'échec ou la réussite de la condition n'est jamais indiqué dans le rapport car il s'agit d'une vérification « muette ».

Les conditions peuvent être du type « **and** » ou « **or** ».

Référence des fichiers de conformité d'audit pour la configuration VMware vCenter/ESXi

Cette section décrit le format et les fonctions des contrôles de conformité VMware vCenter et ESXi, ainsi que les raisons de chaque paramètre.

Nessus peut effectuer l'audit de VMware à l'aide des API natives en extrayant la configuration, puis en exécutant l'audit sur la base des contrôles répertoriés dans le fichier **.audit** associé.

Exigences

Pour exécuter un scan de conformité concluant sur les systèmes VMware, les utilisateurs doivent disposer des éléments suivants :

1. Identifiants administratifs pour VMware vCenter ou ESXi. (Tenable a développé des API pour ESXi [l'interface gratuite qui permet de gérer les machines virtuelles {VM} sur ESX/ESXi] et vCenter [un produit complémentaire payant proposé par VMware pour gérer un ou plusieurs serveurs ESX/ESXi]. Ce plugin peut utiliser les identifiants ESXi ou vCenter pour s'acquitter de sa tâche.)

2. Stratégie d'audit pour les contrôles de conformité VMware vCenter/ESXi.

3. Plugin avec l'ID 64455 (Contrôles de conformité VMware vCenter/ESXi)

Versions prises en charge

Pour le moment, Nessus peut effectuer des audits ESXi 4.x et 5.x, ainsi que vCenter 4.x and 5.

Type de contrôle

La syntaxe pour la fonction VMware **`.audit`** repose dans une large mesure sur les transformations XPATH et XSL pour son exécution.

L'audit VMware prend en charge trois types de contrôles :

AUDIT_VM

Ce type de contrôle permet d'effectuer l'audit des paramètres de machines virtuelles (voir l'Annexe C pour plus d'informations) :

```
<custom_item>
 type: AUDIT_VM
 description: "VM Setting - 'vmsafe.enable = False'"
 xsl_stmt: "<xsl:template match=\"audit:returnval\">"
 xsl_stmt: "<xsl:value-of
       select=\"audit:propSet/audit:val[@xsi:type='VirtualMachineConfigInfo']/audit:na
       me\"/> : vmsafe.enable : <xsl:value-of
       select=\"audit:propSet/audit:val[@xsi:type='VirtualMachineConfigInfo']/audit:ex
       traConfig[audit:key[text()='vmsafe.enable']]/audit:value\"/>."
 xsl_stmt: "</xsl:template>"
 expect: "vmsafe.enable : 0"
</custom_item>
```

AUDIT_ESX

Ce type de contrôle permet d'effectuer l'audit des paramètres de serveur ESX/ESXi :

```
<custom_item>
 type: AUDIT_ESX
 description : "ESX/ESXi Setting - Syslog.global.logDir"
 xsl_stmt: "<xsl:template match=\"audit:returnval\">"
 xsl_stmt: "Syslog.global.logDir = <xsl:value-of
       select=\"audit:propSet/audit:val[@xsi:type='HostConfigInfo']/audit:option[audit
       :key[text()='Syslog.global.logDir']]/audit:value\"/>"
 xsl_stmt: "</xsl:template>"
 expect: "Syslog.global.logDir : /foo/bar"
</custom_item>
```

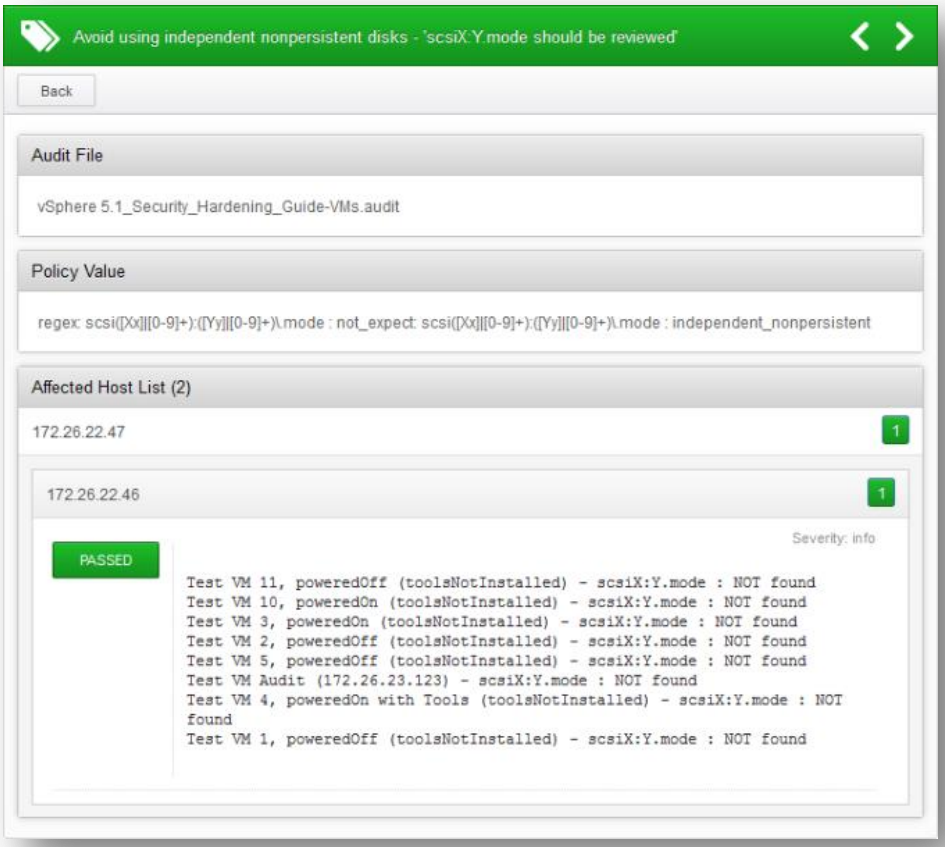
AUDIT_VCENTER

Ce type de contrôle permet d'effectuer l'audit des paramètres vCenter :

```
<custom_item>
 type: AUDIT_VCENTER
 description: "VMware vCenter Setting - config.vpxd.hostPasswordLength"
 xsl_stmt: "<xsl:template match=\"audit:returnval\">"
 xsl_stmt: "config.vpxd.hostPasswordLength = <xsl:value-of
       select=\"audit:propSet/audit:val[@xsi:type='ArrayOfOptionValue']/audit:OptionVal
       ue[audit:key[text()='config.vpxd.hostPasswordLength']]/audit:value\"/>"
```

```
  xsl_stmt: "</xsl:template>"
  expect: "config.vpxd.hostPasswordLength : 30"
</custom_item>
```

Exemple d'audit vSphere réussi :



Mots clés

Le tableau suivant indique comment chaque mot clé peut être utilisé dans les contrôles de conformité de VMware :

Mot clé	Exemple d'utilisation et paramètres pris en charge
type	Ce mot clé décrit le type de contrôle actuellement effectué par un élément donné dans un fichier d'audit. Les audits VMware peuvent être effectués avec les trois types suivants de contrôles d'audit : • AUDIT_VM • AUDIT_ESX • AUDIT_VCENTER
description	Ce mot clé fournit une brève description du contrôle en cours. Il est nécessaire que le champ **description** soit unique et deux contrôles quelconques ne doivent pas avoir le même champ **description**. En effet, SecurityCenter de Tenable utilise ce champ pour produire automatiquement un numéro d'ID de plugin unique basé sur le champ

	description. Exemple : `description: "Disconnect unauthorized devices - 'floppyX.present = false'"`
info	Ce mot clé est utilisé pour ajouter une description plus détaillée au contrôle en cours. Plusieurs champs **info** sont autorisés sans limite prédéfinie. Le contenu du champ info doit toujours être inclus entre guillemets. Exemple : `info: "Make sure floppy drive is not attached"`
regex	Ce mot clé permet de chercher les éléments qui correspondent à une expression regex particulière. Exemple : `regex: "floppy([Xx]|[0-9]+)\\.present :"`

La conformité d'un contrôle peut être déterminée en comparant la sortie du contrôle au mot clé **expect** ou **not_expect**. Vous ne pouvez utiliser qu'une seule balise de test de conformité pour un même contrôle.

Mot clé	Exemple d'utilisation et paramètres pris en charge

expect	Ce mot clé permet de vérifier l'élément de config qui correspond au mot clé **regex**. Si le mot clé **regex** n'est pas utilisé, il recherche la chaîne **expect** dans l'ensemble de config.
	
	Le contrôle renvoie un résultat concluant lorsque la ligne de config trouvée par **regex** correspond à la chaîne **expect** ou, si **regex** n'est pas défini, lorsque la chaîne **expect** est trouvée dans config.
	
	Exemple :
	
	`regex: "floppy([Xx]|[0-9]+)\\.present :"`
	
	`expect: floppy([Xx]|[0-9]+)\\.present : false"`
	
	Ou :
	
	`expect: floppy([Xx]|[0-9]+)\\.present : false"`
	
	Dans les cas ci-dessus, le mot clé **expect** garantit que le lecteur de disquette n'est pas présent.

not_expect	Ce mot clé permet de rechercher les éléments de configuration qui ne doivent pas figurer dans la configuration. Il a l'action contraire de **expect**. Le contrôle est concluant tant que la ligne de config trouvée par **regex** ne correspond pas à la chaîne **not_expect** ou, si le mot clé **regex** n'est pas défini, tant que la chaîne **not_expect** n'est pas trouvée dans la config.

	Exemple :
	
	`regex: floppy([Xx]|[0-9]+)\\.present : "`
	`not_expect: floppy([Xx]|[0-9]+)\\.present : false"`
	
	Ou :
	
	`not_expect: floppy([Xx]|[0-9]+)\\.present : false"`
	
	Dans les cas ci-dessus, le mot clé **`expect`** garantit que le lecteur de disquette n'est pas présent.

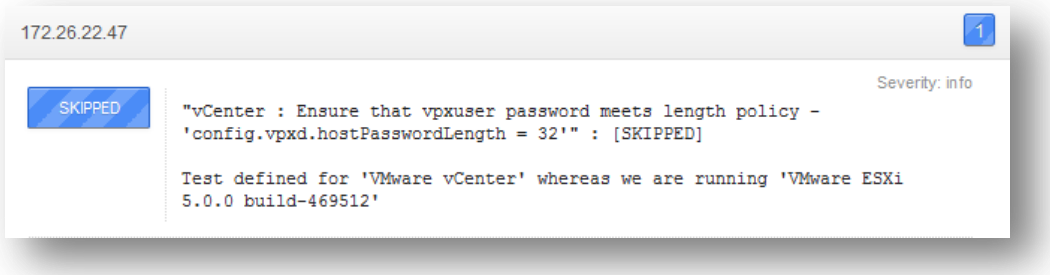
Notes supplémentaires

Si un contrôle est concluant, ce plugin signale toutes les machines virtuelles (VM) qui correspondent à la stratégie. L'audit fourni par Tenable indique à la fois le nom de la machine virtuelle (VM) et l'IP de la cible. Veuillez cependant noter que l'adresse IP d'une machine virtuelle est uniquement disponible lorsque les outils VMware sont installés.

Les rapports se présentent comme suit :

```
Test VM 2, poweredOff (toolsNotInstalled) - vmsafe.enable : NOT found
Test VM Audit (172.26.23.123) - vmsafe.enable : NOT found
```

Vous pouvez utiliser la même stratégie pour scanner ESX/ESXi et vCenter. Veuillez cependant noter que les contrôles vCenter exécutés sur ESX/ESXi seront ignorés.



Pour plus d'informations

Tenable a créé plusieurs autres documents expliquant en détail l'installation, le déploiement, la configuration, l'utilisation et les tests d'ensemble de Nessus.

- **Nessus 5.2 Installation and Configuration Guide** (Guide d'installation et de configuration Nessus 5.2) : explication pas à pas des étapes d'installation et de configuration
- **Nessus 5.2 User Guide** (Guide de l'utilisateur Nessus 5.2) : configuration et utilisation de l'interface utilisateur Nessus
- **Nessus Credential Checks for Unix and Windows** (Contrôles des identifiants Nessus pour Unix et Windows) : informations sur la façon d'effectuer des scans de réseau authentifiés avec le scanner de vulnérabilité Nessus

- **Nessus Compliance Checks** (Contrôles de conformité Nessus) : guide de haut niveau pour comprendre et exécuter les contrôles de conformité au moyen de Nessus et de SecurityCenter
- **Nessus v2 File Format** (Format de fichier Nessus v2) : décrit la structure du format de fichier `.nessus`, qui a été introduit avec Nessus 3.2 et NessusClient 3.2
- **Nessus 5.0 REST Protocol Specification** (Caractéristique du protocole Nessus 5.0 REST) : décrit le protocole REST et l'interface dans Nessus
- **Nessus 5 and Antivirus** (Nessus et les antivirus) : présente le mode d'interaction des progiciels de sécurité les plus courants avec Nessus et fournit des conseils et des solutions qui favoriseront une meilleure coexistence des logiciels, sans compromettre la sécurité ou faire obstacle à vos opérations de scan des vulnérabilités
- **Nessus 5 and Mobile Device Scanning** (Nessus 5 et scan des périphériques mobiles) : décrit comment Nessus s'intègre à Microsoft Active Directory et aux serveurs MDM (serveurs de gestion des périphériques mobiles) afin d'identifier les périphériques mobiles utilisés sur le réseau
- **Nessus 5.0 and Scanning Virtual Machines** (Nessus 5.0 et scan des machines virtuelles) : explique comment utiliser le scanner de vulnérabilité Nessus de Tenable Network Security pour effectuer l'audit de la configuration des plateformes virtuelles et des logiciels exécutés sur ces plateformes
- **Strategic Anti-malware Monitoring with Nessus, PVS, and LCE** (Surveillance stratégique des programmes malveillants avec Nessus, PVS et LCE) : décrit comment la plateforme USM de Tenable peut détecter divers logiciels malveillants, identifier les infections de ces programmes malveillants et en déterminer l'étendue
- **Patch Management Integration** (Intégration de la gestion de correctifs) : décrit comment Nessus et SecurityCenter peuvent tirer parti des identifiants pour les systèmes de gestion de correctif IBM TEM, Microsoft WSUS et SCCM, VMware Go et Red Hat Network Satellite afin d'effectuer l'audit de correctifs sur les systèmes pour lesquels les identifiants peuvent ne pas être mis à la disposition du scanner Nessus
- **Real-Time Compliance Monitoring** (Surveillance de conformité en temps réel) : décrit comment les solutions de Tenable peuvent être utilisées pour faciliter le respect d'un grand nombre de types de règlements gouvernementaux et financiers
- **Tenable Products Plugin Families** (Familles de plugins des produits Tenable) : fournit la description et le résumé des familles de plugins pour Nessus, Log Correlation Engine et Passive Vulnerability Scanner
- **SecurityCenter Administration Guide** (Guide d'administration de SecurityCenter)

D'autres ressources en ligne sont répertoriées ci-dessous :

- Forum de discussions Nessus : https://discussions.nessus.org/
- Blog Tenable : http://www.tenable.com/blog
- Podcast Tenable : http://www.tenable.com/podcast
- Vidéos d'exemples d'utilisation : http://www.youtube.com/user/tenablesecurity
- Feed Twitter Tenable : http://twitter.com/tenablesecurity

N'hésitez pas à contacter Tenable aux adresses support@tenable.com et sales@tenable.com, ou consultez notre site internet sur http://www.tenable.com/.

Annexe A : Exemple de fichier de conformité Unix

Remarque : Le fichier suivant, **`tenable_unix_compliance_template.audit`**, est disponible sur le portail d'assistance de Tenable, à l'adresse https://support.tenable.com/. Ce fichier répertorie les différents types de contrôles de conformité Unix qui peuvent être effectués à l'aide du module de conformité Unix de Tenable. Le fichier en vigueur peut comprendre des mises à jour qui ne sont pas reflétées ici.

```
#
# (C) 2008-2010 Tenable Network Security, Inc.
#
# This script is released under the Tenable Subscription License and
# may not be used from within scripts released under another license
# without authorization from Tenable Network Security, Inc.
#
# See the following licenses for details:
#
# http://cgi.tenablesecurity.com/Nessus_3_SLA_and_Subscription_Agreement.pdf
# http://cgi.tenablesecurity.com/Subscription_Agreement.pdf
#
# @PROFESSIONALFEED@
#
# $Revision: 1.11 $
# $Date: 2010/11/04 15:54:36 $
#
# NAME            : Cert UNIX Security Checklist v2.0
#
#
# Description     : This file is used to demonstrate the wide range of
#             checks that can be performed using Tenable's Unix
#             compliance module. It consists of all the currently
#             implemented built-in checks along with examples of all
#             the other Customizable checks. See:
#             https://plugins-customers.nessus.org/support-
center/nessus_compliance_checks.pdf
#                  For more information.
#
#
###################################
#                         #
# File permission related checks #
#                         #
###################################


<check_type:"Unix">

# Example 1.
# File check example with owner and group
# fields set and mode field set in Numeric
# format

<custom_item>
      #system           : "Linux"
      Type              : FILE_CHECK
      Description       : "Permission and ownership check /etc/inetd.conf"
      Info        : "Checking that /etc/inetd.conf has owner/group of root and is mode
'600'"
      File              : "/etc/inetd.conf"
```

```
      Owner             : "root"
      Group             : "root"
      Mode              : "600"
</custom_item>


# Example 2.
# File check example with just owner field set
# and mode set.

<custom_item>
      #system           : "Linux"
      Type              : FILE_CHECK
      description       : "Permission and ownership check /etc/hosts.equiv"
      info        : "Checking that /etc/hosts.equiv is owned by root and mode '500'"
      file              : "/etc/hosts.equiv"
      owner             : "root"
      mode              : "-r-x------"
</custom_item>

# Example 3.
# File check example with just file field set
# starting with "~". This check will search
# and audit the file ".rhosts" in home directories
# of all accounts listed in /etc/passwd.

<custom_item>
      #system           : "Linux"
      Type              : FILE_CHECK
      description       : "Permission and ownership check ~/.rhosts"
      info        : "Checking that .rhosts in home directories have the specified
ownership/mode"
      file              : "~/.rhosts"
      owner             : "root"
      mode              : "600"
</custom_item>


# Example 4.
# File check example with mode field having
# sticky bit set. Notice the first integer in
# the mode field 1 indicates that sticky bit is
# set. The first integer can be modified to check
# for SUID and SGUID fields. Use the table below
# to determine the first integer field.
#
# 0   000   setuid, setgid, sticky bits are cleared
# 1   001   sticky bit is set
# 2   010   setgid bit is set
# 3   011   setgid and sticky bits are set
# 4   100   setuid bit is set
# 5   101   setuid and sticky bits are set
# 6   110   setuid and setgid bits are set
# 7   111   setuid, setgid, sticky bits are set
```

```
<custom_item>
      #system           : "Linux"
      Type              : FILE_CHECK
      description       : "Permission and ownership check /var/tmp"
      info        : "Checking that /var/tmp is owned by root and mode '1777'"
      file              : "/var/tmp"
      owner       : "root"
      mode        : "1777"
</custom_item>

# Example 5.
# File check example with mode field having
# sticky bit set in textual form and is owned by root.

<custom_item>
      #system           : "Linux"
      Type              : FILE_CHECK
      description       : "Permission and ownership check /tmp"
      info        : "Checking that the /tmp mode has the sticky bit set in textual form
and is owned by root"
      file              : "/tmp"
      owner       : "root"
      mode        : "-rwxrwxrwt"
</custom_item>

###################################
#                                 #
# Service/Process related checks #
#                                 #
###################################

# Example 6.
# Process check to audit if fingerd is turned
# OFF on a given host.

<custom_item>
      #system           : "Linux"
      type        : PROCESS_CHECK
      description       : "Check fingerd process status"
      info        : "This check looks for the finger daemon to be 'OFF'"
      name        : "fingerd"
      status            : OFF
</custom_item>

# Example 7.
# Process check to audit if sshd is turned
# ON on a given host.

<custom_item>
      #system           : "Linux"
      type        : PROCESS_CHECK
      description       : "Check sshd process status"
      info        : "This check looks for the ssh daemon to be 'ON'"
      name        : "sshd"
      status            : ON
</custom_item>
```

```
##############################
#                  #
# File Content related checks #
#                  #
##############################

# Example 8
# File content check to audit if file /etc/host.conf
# contains the string described in the regex field.
#

<custom_item>
      #System            : "Linux"
      type          : FILE_CONTENT_CHECK
      description   : "This check reports a problem if the order is not 'order hosts,bind'
in /etc/host.conf"
      file          : "/etc/host.conf"
      search_locations : "/etc"
      regex         : "order hosts,bind"
      expect            : "order hosts,bind"
</custom_item>

# Example 9
# This is a better example of a file content check. It first looks
# for the string ".*LogLevel=.*" and if it matches it checks whether
# it matches .*LogLevel=9. For example, if the file was to have LogLevel=8
# this check will fail since the expected value is set to 9.
#

<custom_item>
      #System                : "Linux"
      type              : FILE_CONTENT_CHECK
      description             : "This check reports a problem when the log level setting
in the sendmail.cf file is less than the value set in your security policy."
      file              : "sendmail.cf"
      search_locations  : "/etc:/etc/mail:/usr/local/etc/mail"
      regex             : ".*LogLevel=.*"
         expect               : ".*LogLevel=9"
</custom_item>

# Example 10
# With compliance checks you can cause the shell to execute a command
# and parse the result to determine compliance. The check below determines
# whether the version of FreeBSD on the remote system is compliant with
# corporate standards. Note that since we determine the system type using
# the "system" tag, the check will skip if the remote OS doesn't match
# the one specified.

<custom_item>
      system            : "FreeBSD"
      type        : CMD_EXEC
      description : "Make sure that we are running FreeBSD 4.9 or higher"
      cmd         : "uname -a"
      expect            : "FreeBSD (4\.(9|[1-9][0-9])|[5-9]\.)"
</custom_item>
```

```
##################
#            #
# Builtin Checks #
#            #
##################

# Checks that are not customizable are built
# into the Unix compliance check module. Given below
# are the list of all the checks are the performed
# using the builtin functions. Please refer to the
# the Unix compliance checks documentation for more
# details about each check.
#


<item>
name: "minimum_password_length"
description : "Minimum password length"
value : "14..MAX"
</item>

<item>
name: "max_password_age"
description : "Maximum password age"
value: "1..90"
</item>

<item>
name: "min_password_age"
description : "Minimum password age"
value: "6..21"
</item>

<item>
name: "accounts_bad_home_permissions"
description : "Account with bad home permissions"
</item>

<item>
name: "accounts_without_home_dir"
description : "Accounts without home directory"
</item>

<item>
name: "invalid_login_shells"
description: "Accounts with invalid login shells"
</item>

<item>
name: "login_shells_with_suid"
description : "Accounts with suid login shells"
</item>

<item>
name: "login_shells_writeable"
description : "Accounts with writeable shells"
</item>
```

```
<item>
name: "login_shells_bad_owner"
description : "Shells with bad owner"
</item>

<item>
name: "passwd_file_consistency"
description : "Check passwd file consistency"
</item>

<item>
name: "passwd_zero_uid"
description : "Check zero UID account in /etc/passwd"
</item>

<item>
name : "passwd_duplicate_uid"
description : "Check duplicate accounts in /etc/passwd"
</item>

<item>
name : "passwd_duplicate_gid"
description : "Check duplicate gid in /etc/passwd"
</item>

<item>
name : "passwd_duplicate_username"
description : "Check duplicate username in /etc/passwd"
</item>

<item>
name : "passwd_duplicate_home"
description : "Check duplicate home in /etc/passwd"
</item>

<item>
name : "passwd_shadowed"
description : "Check every passwd is shadowed in /etc/passwd"
</item>

<item>
name: "passwd_invalid_gid"
description : "Check every GID in /etc/passwd resides in /etc/group"
</item>

<item>
name : "group_file_consistency"
description : "Check /etc/group file consistency"
</item>

<item>
name: "group_zero_gid"
description : "Check zero GUID in /etc/group"
</item>
```

```
<item>
name: "group_duplicate_name"
description : "Check duplicate group names in /etc/group"
</item>

<item>
name: "group_duplicate_gid"
description : "Check duplicate gid in /etc/group"
</item>

<item>
name : "group_duplicate_members"
description : "Check duplicate members in /etc/group"
</item>

<item>
name: "group_nonexistant_users"
description : "Check for nonexistent users in /etc/group"
</item>

</check_type>
```

Annexe B : Exemple de fichier de conformité Windows

Remarque : Le fichier suivant est disponible sur le portail d'assistance de Tenable, à l'adresse https://support.tenable.com/. Le fichier en vigueur peut comprendre des mises à jour qui ne sont pas reflétées ici. Ce script particulier, **financial_microsoft_windows_user_audit_guideline_v2.audit**, est basé sur les guides courants de renforcement pour l'administration des utilisateurs Cette stratégie recherche une stratégie de mot de passe raisonnable et une stratégie de verrouillage de compte, et elle assure que les événements de connexion sont enregistrés dans le journal des événements Windows.

```
# (C) 2008 Tenable Network Security
#
# This script is released under the Tenable Subscription License and
# may not be used from within scripts released under another license
# without authorization from Tenable Network Security Inc.
#
# See the following licenses for details:
#
# http://cgi.tenablesecurity.com/Nessus_3_SLA_and_Subscription_Agreement.pdf
# http://cgi.tenablesecurity.com/Subscription_Agreement.pdf
#
# @PROFESSIONALFEED@
#
# $Revision: 1.2 $
# $Date: 2008/10/07 15:48:17 $
#
# Synopsis: This file will be read by compliance_check.nbin
#         to check compliance of a Windows host to
#           typical financial institution audit policy
#

<check_type:"Windows" version:"2">
<group_policy:"User audit guideline">

      <item>
      name: "Enforce password history"
      value: 24
      </item>

      <item>
      name: "Maximum password age"
      value: 90
      </item>

      <item>
      name: "Minimum password age"
      value: 1
      </item>

      <item>
      name: "Minimum password length"
      value: [12..14]
      </item>

      <item>
      name: "Account lockout duration"
      value: [15..30]
      </item>
```

```
      <item>
      name: "Account lockout threshold"
      value: [3..5]
      </item>

      <item>
      name: "Reset lockout account counter after"
      value: [15..30]
      </item>

      <item>
 name: "Audit account logon events"
 value: "Success, Failure"
      </item>

 <item>
 name: "Audit logon events"
 value: "Success, Failure"
      </item>

</group_policy>
</check_type>
```

Annexe C : Conversion des transformations XSL vers .audit

Plusieurs plugins de contrôle de conformité reposent sur l'audit du contenu XML, notamment les contrôles de conformité Palo Alto, VMware et Unix. Pour tirer le meilleur parti de ces fonctions, il est recommandé de se familiariser avec la création des transformations XSL. Dans certains cas, la création d'une transformation XSL exige un processus par essais et erreurs. Une fois que vous maîtrisez ce processus, vous pouvez passer à la conversion d'un fichier **.audit** (cette opération n'est pas particulièrement intuitive). Cette annexe propose des conseils qui vous aideront à créer et utiliser des transformations XSL personnalisées, et à les convertir en fichiers **.audit**.

Plusieurs contrôles d'audit (par exemple AUDIT_XML, AUDIT_VCENTER, AUDIT_ESX) sont des opérations séparées et distinctes qui reposent cependant sur la même logique sous-jacente. Si vous comprenez les principes XML de base, vous serez en mesure d'effectuer directement les conversions vers les autres plateformes qui utilisent XML.

L'utilitaire **xsltproc** propose aux utilisateurs une procédure en 7 étapes qui leur permettra de générer des fichiers **.audit** personnalisés pour le contenu XML.

Étape 1 : Installez xsltproc

Assurez-vous que **xsltproc** est installé sur votre système et installez-le si nécessaire. Tapez la commande suivante pour vous assurer qu'il est installé et opérationnel :

```
[tater@pearl ~]# xsltproc
Usage: xsltproc [options] stylesheet file [file ...]
   Options :
        --version or -V: show the version of libxml and libxslt used
        --verbose or -v: show logs of what's happening
[..]
```

Étape 2 : Identifiez le fichier XML à utiliser

Déterminez le fichier XML que vous allez utiliser. Vérifiez l'emplacement du fichier et assez-vous qu'il contient du contenu XML. Par exemple :

```
[tater@pearl ~]# ls top-applications.xml
-rw-r--r-- 1 tater gpigs 3857 2011-09-08 21:20 top-applications.xml
[tater@pearl ~]# head top-applications.xml
<?xml version="1.0"?>
<report reportname="top-applications" logtype="appstat">
  <result name="Top applications" logtype="appstat" start="2013/01/29 00:00:00" start-
       epoch="1359446400" end="2013/01/29 23:59:59" end-epoch="1359532799" generated-
       at="2013/01/30 02:02:09" generated-at-epoch="1359540129" range="Tuesday,
       January 29, 2013">
    <entry>
[..]
```

Étape 3 : Familiarisez-vous avez les transformations XSL et XPath

L'étape suivante exige une compréhension de base des transformations XSL et des concepts XPath. Pour plus d'informations :

- w3schools.com - XSLT – Transformation
- w3schools.com - XPath Introduction

Étape 4 : Créez la transformation XSLT

Cette étape a pour but d'extraire les données pertinentes d'un fichier XML à l'aide des transformations XSL. Commencez par créer une transformation XSL, nécessaire pour extraire les données pertinentes du fichier. Par exemple, supposons

que nous devons extraire l'élément « name » d'un fichier XML. La transformation XSLT permettra d'extraire les informations requises :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"/>

<xsl:template match="result">
<xsl:for-each select="entry">
+ <xsl:value-of select="name"/>
</xsl:for-each>

</xsl:template>
</xsl:stylesheet>
```

Une fois la transformation XSLT créée, enregistrez-la dans un emplacement pratique afin de la tester au cours de l'étape suivante. Cet exemple peut être enregistré sous **pa.xsl**.

Lorsque vous utilisez une transformation XSLT personnalisée dans un fichier .audit, vous devez ignorer les 3 premières lignes et les 2 dernières lignes. Ces lignes standard sont ajoutées par le plugin Nessus **nbin** au cours du traitement. Dans cet exemple, les lignes 5-8 sont celles qui nous intéressent et elles devront être utilisées dans l'élément AUDIT_XML ou AUDIT_REPORTS.

Le processus de test de l'étape 5 peut également être utilisé pendant la création de la transformation XSLT afin de valider les suppositions et/ou les nouvelles techniques. Ce processus est particulièrement utile si vous n'avez pas l'habitude des transformations XSLT ou si vous travaillez sur des transformations plus complexes.

Étape 5 : Confirmez le bon fonctionnement de la transformation XSLT

Vérifiez que votre transformation XSL fonctionne avec xsltproc. Le format général de test est le suivant :

```
/usr/bin/xsltproc {XSLT file} {Source XML}
```

Si vous utilisez les exemples de noms de fichiers provenant des étapes ci-dessus, vous obtenez le résultat suivant. Il vous indique que la transformation XSL est correcte et correctement formatée, et qu'elle renvoie les données recherchées.

```
[tater@pearl ~]# xsltproc pa.xsl top-applications.xml

+ insufficient-data
+ ping
+ snmp
+ dns
+ lpd
+ ntp
+ time
+ icmp
+ netbios-ns
+ radius
+ source-engine
+ stun
+ rip
+ tftp
+ echo
+ portmapper
+ teredo
```

```
+ slp
+ ssdp
+ dhcp
+ mssql-mon
+ pcanywhere
+ apple-airport
+ ike
+ citrix
+ xdmcp
+ l2tp
```

Étape 6 : Copiez les lignes XSLT dans le contrôle .audit

Une fois que la transformation XSL fonctionne comme prévu, copiez les lignes XSLT d'intérêt (les lignes 5-8 dans cet exemple) dans le contrôle **`.audit`**.

```
xsl_stmt: "<xsl:template match=\"result\">"
xsl_stmt: "<xsl:for-each select=\"entry\">"
xsl_stmt: "+ <xsl:value-of select=\"name\"/>"
xsl_stmt: "</xsl:for-each>"
```

Chaque ligne de la transformation XSL personnalisée doit être placée dans son propre élément **`xsl_stmt`**, placée entre guillemets doubles. Puisque l'élément **`xslt_stmt`** utilise les guillemets pour encapsuler les déclarations `<xsl>`, les guillemets doubles doivent être inclus dans une séquence d'échappement. **Il est important d'inclure les guillemets doubles dans une séquence d'échappement et toute omission peut entraîner des erreurs à l'exécution du contrôle**.

```
/usr/bin/xsltproc {XSLT file} {Source XML}
```

L'étape suivante propose plusieurs exemples de guillemets correctement inclus dans une séquence d'échappement.

Étape 7 : Audit final

Une fois que vous avez effectué les six premières étapes, vous disposez de tous les éléments requis pour construire un audit :

```
<custom_item>
 type: AUDIT_REPORTS
 description: "Palo Alto Reports - Top Applications"
 request: "&reporttype=predefined&reportname=top-applications"
 xsl_stmt: "<xsl:template match=\"result\">"
 xsl_stmt: "<xsl:for-each select=\"entry\">"
 xsl_stmt: "+ <xsl:value-of select=\"name\"/>"
 xsl_stmt: "</xsl:for-each>"
</custom_item>
```

À propos de Tenable Network Security

Les solutions de Tenable Network Security sont utilisées par plus de 20 000 organisations, dont tous les services du Département de la Défense des États-Unis (DOD, Department of Defense) ainsi que de nombreuses grandes entreprises internationales et agences gouvernementales, pour prendre une longueur d'avance sur les vulnérabilités, menaces et risques de conformité émergents. Ses solutions Nessus et SecurityCenter continuent de définir la norme pour l'identification des vulnérabilités, la prévention des attaques et le respect de nombreuses exigences règlementaires. Pour plus d'informations, veuillez consulter www.tenable.com.

SIÈGE MONDIAL

Tenable Network Security
7021 Columbia Gateway Drive
Suite 500
Columbia, Maryland 21046
410.872.0555
www.tenable.com