Nessus コンプライアンス チェック リファレンス

2014 年 4 月 23 日

(第 48 版)

目次

はじめに

本書では、Unix、Windows、データベース、SCADA、IBM iSeries、Cisco の各システムのシンタックスをコンプライアンス ポリシーに照らして監査したり、さまざまなシステムのコンテンツに機密コンテンツがないか確認するのに使用されるカスタム **.audit** ファイルを作成するシンタックスについて説明します。

本書は、コンプライアンス監査ファイルのシンタックスを理解して、手動でコンプライアンス監査ファイルを作成できるようにすることを目的としているガイドです。Tenable コンプライアンス チェックがどのように行われるかについては、Tenable サポート ポータルからアクセスできる Nessus コンプライアンス チェック PDF を参照してください。

Nessus は SCADA システム監査をサポートしていますが、この機能は本書の範囲外になるので、この機能の詳細については、Tenable SCADA 情報ページ (こちら) を参照してください。

前提条件

本書は、Nessus 脆弱性スキャナに関してある程度の知識と、監視対象システムに関して詳しい知識を持っている方を対象としています。Unix および Windows のローカル パッチ監査を実行するための Nessus の構成方法については、「Unix と Windows での Nessus 認証チェック」(http://www.tenable.com/products/nessus/documentation) を参照してください。

表記の規則

本書では、ファイル名、デーモン、および実行可能ファイルは、**courier bold** で記載されています。

また、コマンド ライン オプションおよびキーワードも **courier bold** フォントで記載されています。コマンド ラインの例には、コマンド ライン プロンプトおよびコマンド実行の結果得られた出力テキストが含まれる場合と含まれない場合があります。コマンド ラインの例では、実行中のコマンド (ユーザー入力文字) は **courier bold** で、システムによって生成されたサンプル出力は courier (not bold) で記載されています。以下に Unix **pwd** コマンドの実行例を示します。

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

重要な注意点は、この記号と、グレーのテキスト ボックスで示します。

ヒント、例、およびベスト プラクティスは、この記号と、青字に白文字で示します。

Windows 構成監査コンプライアンス ファイル リファレンス

Windows **.audit** コンプライアンス ファイルは、特定書式のテキスト ファイルがベースになっています。ファイル内の項目により、ローカル セキュリティ ポリシー チェックなど汎用的なチェックに加えて、レジストリ設定チェックなど、さまざまな "カスタム項目" チェックを起動できます。本ガイドを通してさまざまな例を紹介します。

引用符の用法:

監査フィールドを囲む一重引用符と二重引用符は、次の 2 つの場合を除いてどちらでも使用可能です。

1. CRLF など、文字をリテラルに解釈する必要がある特殊フィールドを含む Windows コンプライアンス チェックでは、一重引用符を使用します。文字列として解釈する組み込みフィールドはエスケープする必要があります。

たとえば、次のとおりです。

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

2. 二重引用符は WindowsFiles の "include_paths" と "exclude_paths" を使用する際に必要です。

どのフィールドタイプ (description、value_data、regex など) においても、一重引用符または二重引用符を含む文字列を使用する場合は、次のいずれかの方法を用います。

a. 外側の引用符とは異なる引用符タイプを使用する。

例:

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

b. 組み込み引用符はバックスラッシュでエスケープする (二重引用符のみ)。

例:

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

チェック タイプ

Windows コンプライアンス チェックはすべて "Windows" 指定とバージョン "2" が含まれる **check_type** カプセルにより囲む必要があります。

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

「付録 B」で例として紹介されている Windows コンプライアンス チェックは、"Windows" とバージョン "2" の **check_type** 設定で始まり、"**</check_type>**" タグで終わっています。

これは、Windows **.audit** ファイルと、Unix (または他のプラットフォーム) 用のファイルとを区別するために必要です。

値データ

.audit ファイル シンタックスには、チェックをカスタマイズするためのさまざまな値タイプを割り当てることができるキーワードが含まれます。このセクションでは、これらのキーワードと、入力可能なデータのフォーマットについて説明します。

データ型

チェックに入力できるデータ型は次のとおりです。

データ型	説明
DWORD	0 ～ 2,147,483,647
RANGE [X..Y]	X は DWORD または MIN、Y は DWORD または MAX

例:

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

加えて、数値は、"符号" としてプラス記号 (+) またはマイナス記号 (-) を付けて、16 進数値として指定できます。16 進数と符号は組み合わせて使用できます。POLICY_DWORD に対する REGISTRY_SETTING 監査内での有効な例 (括弧内の対応ラベルはなし) を次に示します。

```
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)
```

複雑な表現

次の要素を使用して、**value_data** フィールドに対して複雑な表現を設定できます。

- ||:条件 OR
- &&:条件 AND
- |:バイナリ OR (ビット オペレーション)
- &:バイナリ AND (ビット オペレーション)
- (と):複雑な表現を区切る

例:

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

"check_type" フィールド

このチェック タイプは、一般的な監査タイプ (Windows、WindowsFiles、Unix、Database、Cisco) を示す各監査ファイルの開始部分で使用される前記の "**check_type**" フィールドとは異なります。これは、Windows **value_data** 値に対して実行できるオプションで、実行されるチェック タイプを示します。使用可能な設定は次のとおりです。

- CHECK_EQUAL:リモート値とポリシー値を比較 (**check_type** の指定がない場合のデフォルト設定)
- CHECK_EQUAL_ANY:**value_data** の各要素がシステム リストに少なくとも 1 回登場していることを確認
- CHECK_NOT_EQUAL:リモート値がポリシー値と異なることを確認
- CHECK_GREATER_THAN:リモート値がポリシー値より大きいことを確認
- CHECK_GREATER_THAN_OR_EQUAL:リモート値がポリシー値以上であることを確認
- CHECK_LESS_THAN:リモート値がポリシー値未満であることを確認

- CHECK_LESS_THAN_OR_EQUAL:リモート値がポリシー値以下であることを確認
- CHECK_REGEX:リモート値がポリシー値内の regex と一致していることを確認 (POLICY_TEXT および POLICY_MULTI_TEXT とのみ使用可)
- CHECK_SUBSET:リモート ACL がポリシー ACL のサブセットであることを確認 (ACL とのみ使用可)
- CHECK_SUPERSET:リモート ACL がポリシー ACL のスーパーセットであることを確認 (権限否定 ACL とのみ使用可)

次の監査例では、Guest アカウントに対してアカウント名 "Guest" が存在していないことを確認します。

```
<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>
```

"Guest" 以外の値が存在している場合、監査に通ったことになります。"Guest" が見つかれば、監査に通りません。

"group_policy" フィールド

"**group_policy**" フィールドは、監査を説明する短いテキスト文字列を提供するのに使用できます。**group_policy** は監査ファイルに必須で、**check_type** フィールドの後に挿入する必要があります。

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

…

</group_policy>
</check_type>
```

"info" フィールド

オプションの "**info**" フィールドは、1 つ以上の外部参照による各監査フィールドのラベル付けを行うのに使用できます。たとえば、CIS 固有の監査要件に加えて NIST CCE タグからの参照を配置するのにこのフィールドを使用します。これらの外部参照は、Nessus によって実行される最終監査に出力され、Nessus レポートに表示されたり、SecurityCenter のユーザーインタフェースを介して表示されるなどします。

以下の例は、架空の会社ポリシーの参照が一覧表示されるパスワード監査ポリシーです。

```
<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>
```

1 回の監査に複数のポリシー参照が必要な場合、"**info**" キーワードによって指定される文字列内に、"\n" 区切り記号を使って、複数の文字列を示すことができます。次に例を示します。

```
<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>
```

この監査機能を **nasl** コマンドライン ツールで実行すると、次の出力が生成されます。

```
# /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"
```

"debug" フィールド

オプションの "**debug**" フィールドは、Windows コンテンツ コンプライアンス チェックのトラブルシューティングに使用できます。debug キーワードは、実行されるコンテンツ スキャンについての情報 (処理またはスキャン対象のファイル、結果が検出されたかどうかなど) を出力します。出力量が多いので、このキーワードは、トラブルシューティング目的でのみ使用してください。たとえば、次のとおりです。

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

ACL フォーマット

このセクションでは、ファイルまたはフォルダに目的の ACL 設定が含まれているかどうかを判断するのに使用されるシンタックスについて説明します。

ファイル アクセス制御チェック

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

ファイル ACL (Access Control List) はキーワード **file_acl** で識別されます。ACL 名は、1 つのファイル パーミッション項目に対して一意である必要があります。1 つのファイル ACL に 1 つまたは複数のユーザー エントリを含めることができます。

関連付けられているタイプ	許可されるタイプ
acl_inheritance	• not inherited • inherited • not used
acl_apply	• this folder only • this object only • this folder and files • this folder and subfolders • this folder, subfolders and files • files only • subfolders only • subfolders and files only
acl_allow **acl_deny**	これらの設定はオプションです。 汎用権限: • full control • modify • read & execute • read • write • list folder contents 詳細権限: • full control • traverse folder / execute file • list folder / read data • read attributes • read extended attributes • create files / write data • create folders / append data • write attributes

	• write extended attributes • delete subfolder and files • delete • read permissions • change permissions • take ownership

次に、ファイル アクセス制御 **.audit** テキストの例を示します。

```
<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>
```

レジストリ アクセス制御チェック

使用法

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

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

</acl>
```

レジストリ ACL はキーワード **registry_acl** によって識別されます。ACL 名は、1 つのレジストリ パーミッション項目に対して一意である必要があります。1 つのレジストリ ACL に 1 つまたは複数のユーザー エントリを含めることができます。

関連付けられているタイプ	許可されるタイプ
acl_inheritance	• not inherited • inherited • not used
acl_apply	• this key only • this key and subkeys • subkeys only
acl_allow **acl_deny**	これらの設定はオプションで、ユーザーがオブジェクトに対して持つ権限を定義するのに使用します。 汎用権限: • full control • read 詳細権限: • full control • query value • set value • create subkey • enumerate subkeys • notify • create link • delete • write dac • write owner • read control

次に、レジストリ アクセス制御リスト **.audit** テキストの例を示します。

```
<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>
```

サービス アクセス制御チェック

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

サービス ACL はキーワード **service_acl** によって識別されます。ACL 名は、1 つのサービス パーミッション項目に対して一意である必要があります。1 つのサービス ACL に 1 つまたは複数のユーザー エントリを含めることができます。

関連付けられているタイプ	許可されるタイプ
acl_inheritance	• not inherited • inherited • not used
acl_apply	• this object only
acl_allow **acl_deny**	これらの設定はオプションで、ユーザーがオブジェクトに対して持つ権限を定義するのに使用します。 汎用権限: • full control • read • start, stop and pause • write • delete 詳細権限: • full control • delete • query template • change template

	• query status • enumerate dependents • start • stop • pause and continue • interrogate • user-defined control • read permissions • change permissions • take ownership

サービス アクセス制御チェックの例を次に示します。

```
<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>
```

LaunchPermission コントロール チェック

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

起動 ACL はキーワード **launch_acl** によって識別されます。ACL 名は、1 つの DCOM 起動パーミッション項目に対して一意である必要があります。1 つの起動 ACL に 1 つまたは複数のユーザー エントリを含めることができます。

関連付けられているタイプ	許可されるタイプ
acl_inheritance	• not inherited • inherited
acl_apply	• this object only

acl_allow **acl_deny**	これらの設定はオプションで、ユーザーがオブジェクトに対して持つ権限を定義するのに使用します。 汎用権限: • local launch • remote launch • local activation • remote activation

この ACL は、Windows XP/2003/Vista (および一部 Windows 2000) に対してのみ有効です。

起動アクセス制御チェックの例を次に示します。

```
<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>
```

Launch2Permission コントロール チェック

使用法

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

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

</acl>
```

launch2 ACL はキーワード **launch2_acl** によって識別されます。ACL 名は、1 つの DCOM 起動パーミッション項目に対して一意である必要があります。1 つの launch2 ACL に 1 つまたは複数のユーザー エントリを含めることができます。

関連付けられているタイプ	許可されるタイプ
acl_inheritance	• not inherited • inherited
acl_apply	• this object only
acl_allow **acl_deny**	これらの設定はオプションで、ユーザーがオブジェクトに対して持つ権限を定義するのに使用します。 汎用権限: • launch

launch2 ACL は Windows 2000 システムと NT システムに対してのみ使用できます。

起動アクセス制御チェックの例を次に示します。

```
<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>
```

アクセス パーミッション コントロール チェック

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

アクセス ACL はキーワード **access_acl** によって識別されます。ACL 名は、1 つの DCOM アクセス パーミッション項目に対して一意である必要があります。1 つのアクセス ACL に 1 つまたは複数のユーザー エントリを含めることができます。

関連付けられているタイプ	許可されるタイプ
acl_inheritance	• not inherited • inherited
acl_apply	• this object only
acl_allow **acl_deny**	これらの設定はオプションで、ユーザーがオブジェクトに対して持つ権限を定義するのに使用します。 汎用権限: • local access • remote access

起動アクセス制御チェックの例を次に示します。

```
<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>
```

カスタム項目

カスタム項目は、前記のキーワードに基づいて定義される 1 つの完全なチェックです。次に、使用可能なカスタム項目のタイプを示します。各チェックが "**<custom_item>**" タグで始まり、"**</custom_item>**" で終わります。これらのタグ内に、コンプライアンス チェック パーサーによって解釈され、チェックを実行する 1 つ以上のキーワードが配置されます。

カスタム監査チェックでは、終了タグとして "**</custom_item>**" と "**</item>**" のどちらでも使用できます。

PASSWORD_POLICY

使用法

```
<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>
```

このポリシー項目は、"Windows の設定 -> セキュリティの設定 -> アカウント ポリシー -> パスワードのポリシー" で定義されている値に対するチェックを行います。

このチェックは、関数 **NetUserModalsGet** をレベル 1 で呼び出すことにより実行されます。

この項目は、**password_policy** フィールドを使用して、監査の必要があるパスワード ポリシーの要素を示します。許可されるタイプは次のとおりです。

- ENFORCE_PASSWORD_HISTORY ("パスワードの履歴を記録する")
  `value_type: POLICY_DWORD`
  `value_data: DWORD or RANGE [number of remembered passwords]`
- MAXIMUM_PASSWORD_AGE ("パスワードの有効期間")
  `value_type: TIME_DAY`
  `value_data: DWORD or RANGE [time in days]`
- MINIMUM_PASSWORD_AGE ("パスワードの変更禁止期間")
  `value_type: TIME_DAY`
  `value_data: DWORD or RANGE [time in days]`
- MINIMUM_PASSWORD_LENGTH ("パスワードの長さ")
  `value_type: POLICY_DWORD`
  `value_data: DWORD or RANGE [minimum number of characters in the password]`
- COMPLEXITY_REQUIREMENTS ("パスワードは要求する複雑さを満たす")
  `value_type: POLICY_SET`
  `value_data: "Enabled" or "Disabled"`

- REVERSIBLE_ENCRYPTION ("暗号化を元に戻せる状態でドメインのすべてのユーザーのパスワードを保存する")
  value_type: POLICY_SET
  value_data: "Enabled" or "Disabled"

- FORCE_LOGOFF ("ネットワーク セキュリティ:ログオン時間を経過した場合はユーザーを強制的にログオフさせる")
  value_type: POLICY_SET
  value_data: "Enabled" or "Disabled"

ポリシー "暗号化を元に戻せる状態でドメインのすべてのユーザーのパスワードを保存する" に対するチェック方法は現在のところありません。

FORCE_LOGOFF ポリシーは "セキュリティの設定 -> ローカル ポリシー -> セキュリティ オプション" で確認できます。

次に、パスワード ポリシー監査例を示します。

```
 <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

使用法

```
<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>
```

このポリシー項目は、"セキュリティの設定 -> アカウント ポリシー -> アカウント ロックアウトのポリシー" で定義されている値に対するチェックを行います。

このチェックは、関数 **NetUserModalsGet** をレベル 3 で呼び出すことにより実行されます。

この項目は、**lockout_policy** フィールドを使用して、監査の必要があるアカウント ロックアウト ポリシーの要素を示します。許可されるタイプは次のとおりです。

- LOCKOUT_DURATION ("ロックアウト期間")
  value_type: TIME_MINUTE
  value_data: DWORD or RANGE [time in minutes]

- LOCKOUT_THRESHOLD ("アカウントのロックアウトのしきい値")
  value_type: POLICY_DWORD
  value_data: DWORD or RANGE [time in days]

- LOCKOUT_RESET ("ロックアウト カウンタのリセット")
  value_type: TIME_MINUTE
  value_data: DWORD or RANGE [time in minutes]

例:

```
 <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

使用法

```
<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>
```

このポリシー項目は、"セキュリティの設定 -> アカウント ポリシー -> Kerberos ポリシー" で定義されている値に対するチェックを行います。

このチェックは、関数 **NetUserModalsGet** をレベル 1 で呼び出すことにより実行されます。

この項目は、**kerberos_policy** フィールドを使用して、監査の必要がある Kerberos ポリシーの要素を示します。許可されるタイプは次のとおりです。

- USER_LOGON_RESTRICTIONS ("ユーザー ログオンの制限を強制する")
  value_type: POLICY_SET
  value_data: "Enabled" or "Disabled"
- SERVICE_TICKET_LIFETIME ("サービス チケットの最長有効期間")
  value_type: TIME_MINUTE
  value_data: DWORD or RANGE [time in minutes]
- USER_TICKET_LIFETIME ("チケットの最長有効期間")
  value_type: TIME_HOUR
  value_data: DWORD or RANGE [time in hours]
- USER_TICKET_RENEWAL_LIFETIME ("ユーザーチケットを更新できる最長有効期間")
  value_type: TIME_DAY
  value_data: DWORD or RANGE [time in day]
- CLOCK_SYNCHRONIZATION_TOLERANCE ("コンピュータの時計の同期の最長トレランス")
  value_type: TIME_MINUTE
  value_data: DWORD or RANGE [time in minute]

Kerberos ポリシーは、Windows では通常ドメイン コントローラである KDC (キー配布センター) に対してのみチェックできます。

例:

```
 <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

使用法

```
<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>
```

このポリシー項目は、"セキュリティの設定 -> ローカル ポリシー -> 監査ポリシー" で定義されている値に対するチェックを行います。

このチェックは、関数 **LsaQueryInformationPolicy** をレベル **PolicyAuditEventsInformation** で呼び出すことにより実行されます。

この項目は、**audit_policy** フィールドを使用して、監査の必要がある監査ポリシーの要素を示します。許可されるタイプは次のとおりです。

- AUDIT_ACCOUNT_LOGON ("アカウント ログオン イベントの監査")
- AUDIT_ACCOUNT_MANAGER ("アカウント管理の監査")
- AUDIT_DIRECTORY_SERVICE_ACCESS ("ディレクトリ サービスのアクセスの監査")
- AUDIT_LOGON ("ログオン イベントの監査")
- AUDIT_OBJECT_ACCESS ("オブジェクト アクセスの監査")
- AUDIT_POLICY_CHANGE ("ポリシーの変更の監査")
- AUDIT_PRIVILEGE_USE ("特権使用の監査")
- AUDIT_DETAILED_TRACKING ("プロセス追跡の監査")
- AUDIT_SYSTEM ("システム イベントの監査")

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

"Success, Failure" 内には半角スペースが含まれていることに注意してください。

例:

```
 <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

使用法

```
<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>
```

このポリシー項目は、auditpol/get/category:* 内の値に対してチェックを行います。

このチェックは、WMI から cmd.exe auditpol/get/category:* を実行することにより実施されます。

この項目は、**audit_policy_subcategory** フィールドを使用して、監査の必要があるサブカテゴリを決定します。許可される SUBCATEGORY_POLICY_TYPE は次のとおりです。

- Security State Change
- Security System Extension
- System Integrity
- IPsec Driver
- Other System Events
- Logon
- Logoff
- Account Lockout
- IPsec Main Mode
- IPsec Quick Mode
- IPsec Extended Mode
- Special Logon
- Other Logon/Logoff Events
- Network Policy Server
- File System
- Registry

- Kernel Object
- SAM
- Certification Services
- Application Generated
- Handle Manipulation
- File Share
- Filtering Platform Packet Drop
- Filtering Platform Connection
- Other Object Access Events
- Sensitive Privilege Use
- Non Sensitive Privilege Use
- Other Privilege Use Events
- Process Creation
- Process Termination
- DPAPI Activity
- RPC Events
- Audit Policy Change
- Authentication Policy Change
- Authorization Policy Change
- MPSSVC Rule-Level Policy Change
- Filtering Platform Policy Change
- Other Policy Change Events
- User Account Management
- Computer Account Management
- Security Group Management
- Distribution Group Management
- Application Group Management
- Other Account Management Events
- Directory Service Access
- Directory Service Changes
- Directory Service Replication
- Detailed Directory Service Replication
- Credential Validation
- Kerberos Service Ticket Operations
- Other Account Logon Events

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

"Success, Failure" 内には半角スペースが含まれていることに注意してください。

このチェックは、Windows Vista/2008 Server 以上に対してのみ実行可能です。ファイアウォールが有効な場合、ファイアウォール設定の例外として WMI を追加するだけでなく、**gpedit.msc** を使用してファイアウォール設定の "Windows ファイアウォール：着信リモート管理の例外を許可する" を有効にする必要もあります。このチェックは、英語版以外の Vista/2008 システムまたは auditpol がインストールされていないシステムでは実行されない可能性があります。

例：

```
<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

使用法

```
<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>
```

このチェックでは、リモート サーバー上で "**powershell_args**" で指定される引数により、**powershell.exe** が実行されます。"**only_show_cmd_output**" が YES に設定されている場合は、コマンド出力が返され、**value_data** が指定されている場合は、"**value_data**" と結果との比較が行われます。

関連付けられているタイプ:

この項目は、フィールド "**powershell_args**" を使用して、**powershell.exe** に提供する引数を指定します。powershell.exe の保存場所がデフォルトではない場合、**powershell_console_file** キーワードを使用して、保存場所を指定する必要があります。現在サポートされているのは "get-" コマンドレットのみです。たとえば、次のとおりです。

- `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`

この項目で、コマンド出力全体をレポートする必要がある場合は、オプション フィールド "**only_show_cmd_output**" を使用します。

- `only_show_cmd_output: YES` or `NO`

その他の考慮事項:

1. "**only_show_cmd_output**" を設定して、出力の重大度を設定したい場合、重大度タグを使用して重大度を変更できます。デフォルト値は INFO です。
2. Windows オペレーティング システム (XP、2003 など) の一部、および、このチェックでは結果が得られないシステムには、デフォルトで Powershell はインストールされません。したがって、このチェックを使用する前に、Powershell がリモート ターゲットにインストールされていることを確認する必要があります。
3. このチェックが正しく機能するには、WMI サービスが有効になっている必要があります。さらに、ファイアウォールが "着信リモート管理の例外を許可する" に設定されている必要があります。
4. コマンドレット エイリアス (たとえば "Get-Process" ではなく "gps") は使用できません。

例:

この例では、Powershell コマンドレット "Get-Hotfix" が実行され、ID が "File 1" の修正プログラムは選択しないように where-object が指定され、Description、HotfixID、Installedby がリスト形式でレポートされます。

```
<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>
```

例:

この例では、Windows サービス "WinRM" が実行しているかどうかのチェックが行われます。

```
<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

使用法

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

```
 value_data: "[FILE HASH]"
</custom_item>
```

このチェックでは、リモート サーバー上で、システム上のファイル ハッシュと期待されるファイル ハッシュとの比較を行うための情報とともに **powershell.exe** が実行されます。

その他の考慮事項:

- デフォルトでは、ファイルの MD5 ハッシュが比較されますが、SHA1、SHA256、SHA384、SHA512、RIPEMD160 のアルゴリズムで生成されたハッシュで比較を行うことができます。
- このチェックを行うには、ターゲット上で PowerShell がインストールされ、WMI が有効になっている必要があります。

例:

この例では、指定の MD5 ハッシュと `C:\test\test2.zip` のファイル ハッシュとの比較が行われます。

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

例:

この例では、指定の SHA1 ハッシュと `C:\test\test3.zip` のファイル ハッシュとの比較が行われます。

```
<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

使用法

```
<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>
```

このチェックでは、IIS を実行しているサーバー上で、"**appcmd_args**" で指定されている引数により **appcmd.exe** が実行され、出力と **value_data** を比較して、コンプライアンスが判断されます。場合によっては (構成の一覧表示など)、コマンド出力のレポートが必要な場合があります。そのような場合には、"**only_show_cmd_output**" を使用します。

このチェックは、Windows 上の Internet Information Services (IIS) 7 に対してのみ実行可能です。

この項目は、フィールド "**appcmd_args**" を使用して、**appcmd.exe** に提供する引数を指定します。現在は "list" コマンドしか指定できません。

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

この項目で、コマンド出力全体をレポートする必要がある場合は、オプション フィールド "**only_show_cmd_output**" を使用します。

例:

このチェックでは、"`appcmd.exe list AppPools /processModel.identityType:ApplicationPoolIdentity`" の結果と **value_data** とが比較され、出力に 'APPPOOL "DefaultAppPool"' が含まれていた場合のみ合格となります。

```
<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

使用法

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

このチェックでは、ターゲット上のオープン TCP/UDP ポートのリストが照会され、ポートの許可リストと比較されます。このチェックは、"netstat -ano" または "netstat -an" からの出力に依存してオープン ポート リストを取得し、(get_port_state()/get_udp_port_state()) を使用してポート状態を検証して、これらのポートが本当にオープン ポートかどうかを確認します。

考慮事項:

- **value_data** ではポート範囲として正規表現も使用可能なので、8[0-9]+ などの値も設定できます。

例:

次の例では、ターゲット上のオープン TCP ポート リストと "**value_data**" との比較が行われます。

```
<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>
```

次の例では、ターゲット上のオープン UDP ポート リストと "**value_data**" との比較が行われます。

```
<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

使用法

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

このチェックでは、ターゲット上のオープン TCP/UDP ポートのリストが照会され、ポートの拒否リストと比較されます。このチェックは、"netstat -ano" または "netstat -an" からの出力に依存してオープン ポート リストを取得し、(get_port_state()/get_udp_port_state()) を使用してポート状態を検証して、これらのポートが本当にオープン ポートかどうかを確認します。

許可されるタイプは次のとおりです。

- `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 or UDP`

考慮事項:

- **value_data** ではポート範囲として正規表現も使用可能なので、8[0-9]+ などの値も設定できます。

例:

次の例では、ターゲット上のオープン TCP ポート リストと "**value_data**" との比較が行われます。

```
<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>
```

次の例では、ターゲット上のオープン UDP ポート リストと "**value_data**" との比較が行われます。

```
<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

使用法

```
<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>
```

このチェックでは、指定ポートで実行中のプロセスが照会されます。このチェックは、"netstat -ano" および "tasklist /svc" の出力に依存して、どの TCP/UDP でどのプロセスが実行しているかを判断します。

許可されるタイプは次のとおりです。

- `value_type: POLICY_TEXT`
- `value_data`: Arbitrary string, e.g.,`"foo.exe"`
- `port_type: TCP` or `UDP`
- `port_no`: port number, e.g., 80, 445
- `port_option: CAN_BE_CLOSED`

考慮事項:

- **port_option** が CAN_BE_CLOSED に設定されている場合、そのポートがリモート システムでオープンでなければ合格を返し、オープンであればエラーを返します。
- Windows 2000 およびそれ以前においては、"netstat –ano" がサポートされていないので、このチェックは Windows XP 以降に対してのみ実行できます。

例:

次の例では、TCP ポート 5900 で実行しているプロセスが "vss.exe" または "vssrvc.exe" であることをチェックします。

```
<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>
```

次の例も最初の例と似ていますが、この例では、check_type の使用が含まれています。

```
<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>
```

AUDIT_USER_TIMESTAMPS

使用法

```
<custom_item>
 type: AUDIT_USER_TIMESTAMPS
 description: "Users not logged in past 7 or more days."
 value_type: POLICY_DAY
 value_data: "7"
 timestamp: "LogonTime"
 ignore_users: "Admin*,foo"
 check type: CHECK GREATER THAN OR EQUAL
</custom_item>
```

このチェックでは、ユーザー タイムスタンプを確認して非アクティブのアカウントを照会します。

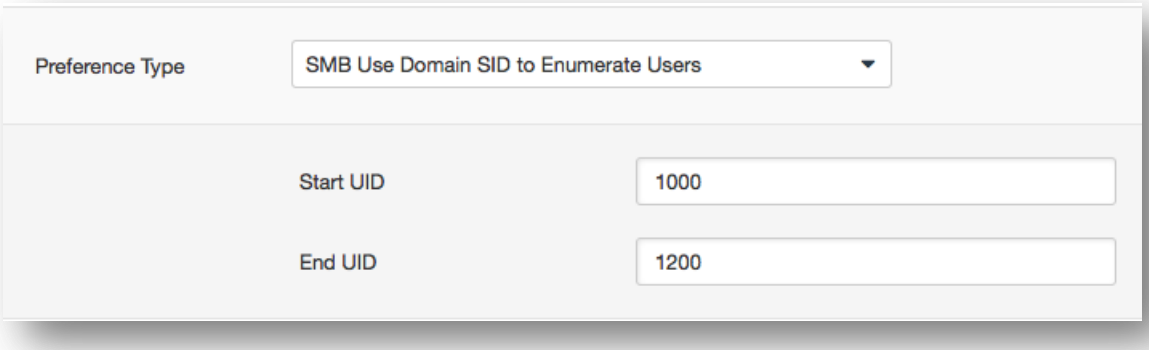
キーワード timestamp で使用できる値は次のとおりです。

- LogonTime

- `LogoffTime`
- `KickoffTime`
- `PassLastSet`
- `PassCanChange`
- `PassMustChange`
- `ACB`

考慮事項:

- デフォルトでは、無効なアカウント、またはパスワードを変更できなかったり、パスワードの有効期間がないアカウントは結果から除外されます。次のように設定することにより、これらを結果に含めることもできます。`include_users: "password never expires" || "cannot change password" || "disabled"`
- デフォルトでは、SID 付きのユーザーは *'SMB Use Host SID to Enumerate Local Users/SMB Use Domain SID to Enumerate Users'* プリファレンス範囲内のユーザーに限られます。



例:

このチェックには、**ignore_users** 指示子により特定のユーザーが結果から除外される機能もあります。

```
<custom_item>
 type: AUDIT_USER_TIMESTAMPS
```

```
  description: "Password not changed in last 90 days"
  value_type: POLICY_DAY
  value_data: "90"
  timestamp: "PassLastSet"
 ignore_users: "Admin*,foo"
  check_type: CHECK_GREATER_THAN_OR_EQUAL
 </custom_item>
```

CHECK_ACCOUNT

使用法

```
<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>
```

このポリシー項目は、"セキュリティの設定 -> ローカル ポリシー -> セキュリティ オプション" で定義されている次の値に対するチェックを行います。

- アカウント:Administrator アカウントの状態
- アカウント:Guest アカウントの状態
- アカウント:Administrator アカウント名の変更
- アカウント:Guest アカウント名の変更

このチェックは、ドメイン/システム SID を取得するためにはレベル **`PolicyAccountDomainInformation`** で、Administrator 名と Guest 名を取得するためには **`LsaLookupSid`** で、アカウント情報を取得するためには **`NetUserGetInfo`** で関数 **`LsaQueryInformationPolicy`** を実行します。

この項目は、**`account_type`** フィールドを使用して、監査の必要があるアカウントを指定します。許可されるタイプは次のとおりです。

- ADMINISTRATOR_ACCOUNT ("アカウント:Administrator アカウントの状態")
  ```
  value_type: POLICY_SET
  value_data: "Enabled" or "Disabled"
  ```
- GUEST_ACCOUNT ("アカウント:Guest アカウントの状態")
  ```
  value_type: POLICY_SET
  value_data: "Enabled" or "Disabled"
  ```
- ADMINISTRATOR_ACCOUNT ("アカウント:Administrator アカウント名の変更")
  ```
  value_type: POLICY_TEXT
  value_data: "TEXT HERE" [administrator name]
  check_type: [CHECK_TYPE] (any one of the possible check_type values)
  ```
- GUEST_ACCOUNT ("アカウント:Guest アカウント名の変更")
  ```
  value_type: POLICY_TEXT
  ```

```
value_data: "TEXT HERE" [guest name]
check_type: [CHECK_TYPE] (any one of the possible check_type values)
```

ドメイン資格情報に依存して、ローカル システム アカウントまたはドメイン アカウントがチェックされます。

例:

```
<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

使用法

```
<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>
```

このポリシー項目は、**lusmgr.msc** 内のグループの名前とステータスをチェックします。

この項目は、**group_type** フィールドを使用して、監査の必要があるグループを指定します。許可されるタイプは次のとおりです。

- ADMINISTRATORS_GROUP
- USERS_GROUP

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

value_type フィールドで許可されるタイプは次のとおりです。

- POLICY_SET (グループのステータスをチェック)
  `value_type: POLICY_SET`
  `value_data: "Enabled" or "Disabled"`
- POLICY_TEXT (グループ名をチェック)
  `value_type: POLICY_TEXT`
  `value_data: "Guests1"` (この場合、**value_data** には任意のテキスト文字列が入ります)

例:

```
<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

使用法

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

このポリシー項目は、"セキュリティの設定 -> ローカル ポリシー -> セキュリティ オプション -> ネットワーク アクセス:匿名の SID と名前の変換を許可する" で定義されている次の値に対するチェックを行います。このチェックは、LSA ポリシー ハンドルで関数 **LsaQuerySecurityObject** を呼び出すと実行されます。

許可されるタイプは次のとおりです。

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

この監査を使用する際には、このポリシーについて次の点に注意してください。

- LSA サービス上でのパーミッション チェックである
- ANONYMOUS_USER にフラグ POLICY_LOOKUP_NAMES が設定されているかどうかをチェックする
- Windows 2003 では廃止 (Windows 2003 では匿名ユーザーが LSA パイプにアクセスできなくなっているため)

例:

```
 <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

このチェックでは、リモート Windows システムが適切に機能するのにリモート レジストリ アクセスを必要とします。

使用法

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

```
</custom_item>
```

このポリシー項目は、"システム サービス" で定義されているスタートアップ値に対するチェックを行います。このチェックは、関数 **RegQueryValueEx** を次のキーで呼び出すことにより実行されます。

- キー:"SYSTEM\CurrentControlSet\Services\" + service_name
- 項目:"Start"

許可されるタイプは次のとおりです。

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

service_name フィールドは、サービスの実際の名前になります。この名前は以下の場合に取得されます。

1. サービス コントロール パネル (管理ツール内) を起動したとき
2. 目的のサービスを選択したとき
3. プロパティ ダイアログ ボックスを開くとき (右クリックからプロパティを選択したとき)
4. "サービス名" 部分を抽出したとき

サービス パーミッション設定は SERVICE_PERMISSIONS 項目で確認できます。

例:

```
 <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

使用法

```
<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>
```

このポリシー項目は、特定のユーザー リストが 1 つ以上のグループに存在していることをチェックします。

許可されるタイプは次のとおりです。

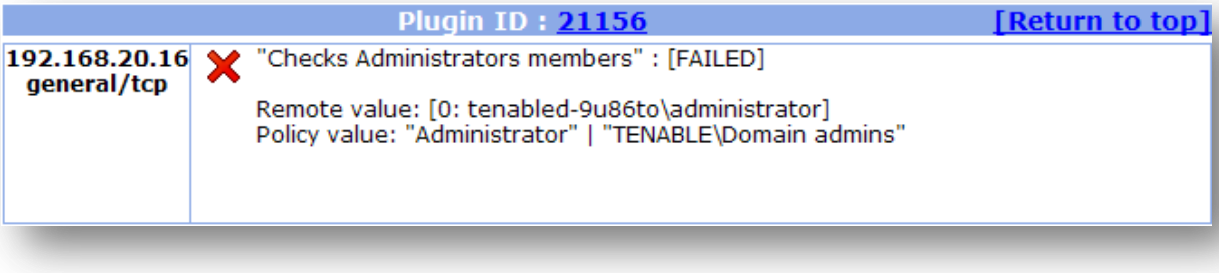
```
value_type: POLICY_TEXT or POLICY_MULTI_TEXT
value_data: "user1"&&"user2"&& ... &&"usern"
```

この監査を使用する際には、ユーザー名は "MYDOMAIN\John Smith" のようなドメイン名で指定できること、および **group_name** フィールドには監査対象の単一のグループを指定するということに注意してください。

単一の Nessus **.audit** ファイルで、複数の顧客項目を指定できるので、複数のグループ内のユーザー リストを簡単に監査できます。次の例は、"Administrators" グループに "Administrator" と "TENABLE\Domain admins" ユーザーのみが含まれていることを確認する **.audit** ポリシーです。

```
<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>
```

次の画像は、上記の **.audit** ファイルを Windows 2003 サーバー上で実行したときのスクリーン キャプチャ例です。



USER_GROUPS_POLICY

使用法

```
<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>
```

このポリシー項目は、Windows ユーザーが **value_data** で指定されているグループに属しているかどうかをチェックします。この監査では、ドメイン ユーザーをドメイン コントローラに対してのみテストできます。"Local Service" などのビルトイン ユーザーに対してはチェックできません。

例:

```
 <custom_item>
  type: USER_GROUPS_POLICY
  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

使用法

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

このポリシー項目は、"セキュリティの設定 -> ローカル ポリシー -> ユーザー権利の割り当て" で定義されている次の値に対するチェックを行います。このチェックは、LSA ポリシー ハンドルで関数 **LsaEnumerateAccountsWithUserRight** を呼び出すと実行されます。

right_type フィールドは監査対象の権限を示します。許可される値は次のとおりです。

```
right_type: RIGHT
```

RIGHT には、以下の権限を指定できます。

```
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
```

許可されるタイプは次のとおりです。

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

ユーザー権限テストは、ドメイン コントローラに対して多くのリクエストを実行します。これらのテストは別途 1 つのポリシー ファイルに含めて、ドメイン コントローラと 1 つのドメイン システムに対してのみ起動する必要があります。

right タイプは、トークンとして解釈されるので、引用符で囲む必要はありません。

例:

```
<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

このチェックでは、リモート Windows システムが適切に機能するのにリモート レジストリ アクセスを必要とします。

使用法

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

このポリシー項目は、ファイル (**value_data**) が存在しているかどうか (**file_option**) をチェックします。このチェックは、関数 **CreateFile** を呼び出すことにより実行されます。

許可されるタイプは次のとおりです。

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

例:

```
 <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 thatbad.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

このチェックでは、リモート Windows システムが適切に機能するのにリモート レジストリ アクセスを必要とします。

使用法

```
<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>
```

このポリシー項目は、**file** フィールドで指定されているファイルのバージョンがリモート ファイル バージョン以上であるかどうかをチェックします (デフォルト)。**check_type** オプションを使用して、リモート ファイル バージョンより大きいかどうかを判断することもできます。

許可されるタイプは次のとおりです。

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

例:

```
 <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

このチェックでは、リモート Windows システムが適切に機能するのにリモート レジストリ アクセスを必要とします。

使用法

```
<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>
```

このポリシー項目は、FILE_PERMISSIONS ACL が正しいかどうかをチェックします。このチェックは、ファイル ハンドルで関数 **GetSecurityInfo** をレベル 7 により呼び出すと実行されます。

許可されるタイプは次のとおりです。

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

次の事前定義済みパスをファイル名またはフォルダ名で使用できます。

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

この監査を使用する際には、次の点に注意してください。

- **file** フィールドには、ファイル名またはフォルダ名の完全パス (**C:\WINDOWS\SYSTEM32** など) を含めることも、上記のようなパス キーワードを含めることもできます。パス キーワードを使用する際には、Nessus がパス可変値を判断できるようにリモート レジストリが有効になっている必要があります。
- **value_data** フィールドは、ポリシー ファイルで定義されている ACL の名前になります。
- **acl_option** フィールドは CAN_BE_NULL または CAN_NOT_BE_NULL に設定でき、ファイルが存在していない場合の合格/不合格の判断に使用されます。

例:

```
<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>
```

上記のチェックが実行されると、コンプライアンス モジュールは、"**%SystemRoot%\SYSTEM32**" に対して定義されているパーミッションが file_acl ACL1 に示されているパーミッションと一致しているかどうかをチェックします。

FILE_AUDIT

このチェックでは、リモート Windows システムが適切に機能するのにリモート レジストリ アクセスを必要とします。

使用法

```
<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>
```

このポリシー項目は、指定の ACL を使用するファイルまたはフォルダの監査プロパティ (プロパティ -> セキュリティ -> 詳細 -> 監査) をチェックします。このチェックは、ファイル ハンドルで関数 **GetSecurityInfo** をレベル SACL_SECURITY_INFORMATION により呼び出すと実行されます。

許可されるタイプは次のとおりです。

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

次の事前定義済みパスをファイル名またはフォルダ名で使用できます。

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

この監査を使用する際には、次の点に注意してください。

- **file** フィールドには、ファイル名またはフォルダ名の完全パス (**C:\WINDOWS\SYSTEM32** など) を含めることも、上記のようなパス キーワードを含めることもできます。パス キーワードを使用する際には、Nessus がパス可変値を判断できるようにリモート レジストリが有効になっている必要があります。
- **value_data** フィールドは、ポリシー ファイルで定義されている ACL の名前になります。
- **acl_option** フィールドは CAN_BE_NULL または CAN_NOT_BE_NULL に設定でき、ファイルが存在していない場合の合格/不合格の判断に使用されます。
- **acl_allow** フィールドと **acl_deny** フィールドは、"Successful" 監査イベントと "Failed" 監査イベントに対応します。

次の例は、"ACL1" というアクセス制御リスト ルールを含む FILE_AUDIT 関数を実装する **.audit** ファイルです。

```
<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

このチェックでは、リモート Windows システムが適切に機能するのにリモート レジストリ アクセスを必要とします。

使用法

```
<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>
```

このポリシー項目は、ファイルに正規表現 **regex** が含まれ、この正規表現が **expect** と一致しているかどうかをチェックします。

このチェックは、ファイル ハンドルで関数 **ReadFile** を呼び出すと実行されます。

ファイルは SMB から、Nessus サーバー上のメモリ バッファに読み込まれ、バッファがコンプライアンス チェック用に処理されます。ファイルは Nessus サーバーのディスク上には保存されません。分析用にメモリ バッファにコピーされるだけです。

許可されるタイプは次のとおりです。

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

次の事前定義済みパスをファイル名またはフォルダ名で使用できます。

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

この監査タイプを使用する際には、次の点に注意してください。

- **value_data** フィールドには、ファイル名またはフォルダ名の完全パス (**C:\WINDOWS\SYSTEM32** など) を含めることも、上記のようなパス キーワードを含めることもできます。パス キーワードを使用する際には、Nessus がパス可変値を判断できるようにリモート レジストリが有効になっている必要があります。
- **regex** フィールドは、この項目がファイルに存在していることをチェックします。
- **expect** フィールドは、この項目が正規表現と一致することをチェックします。
- **file_option** フィールドを CAN_BE_NULL に設定すれば、ファイルが存在していない場合でも合格にできます。
- **file_option** フィールドを CAN_NOT_BE_NULL に設定すると、ファイルが存在して空の場合、不合格になります。
- **avoid_floppy_access** フィールドを設定すると、フロッピー ドライブにアクセスするチェックは回避できます。監査によって、ディスクがないフロッピー ドライブへのアクセスが発生する場合、このフィールドを使用する必要があります。

例:

```
<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

このチェックでは、リモート Windows システムが適切に機能するのにリモート レジストリ アクセスを必要とします。

使用法

```
<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>
```

このポリシー項目は、ファイルに正規表現 regex が含まれ、この正規表現が expect と一致していないことをチェックします。このチェックは、ファイル ハンドルで関数 **ReadFile** を呼び出すと実行されます。

許可されるタイプは次のとおりです。

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

次の事前定義済みパスをファイル名またはフォルダ名で使用できます。

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

この監査タイプを使用する際には、次の点に注意してください。

- **value_data** フィールドには、ファイル名またはフォルダ名の完全パス (**C:\WINDOWS\SYSTEM32** など) を含めることも、上記のようなパス キーワードを含めることもできます。パス キーワードを使用する際には、Nessus がパス可変値を判断できるようにリモート レジストリが有効になっている必要があります。
- **regex** フィールドは、この項目がファイルに存在していることをチェックします。
- **expect** フィールドは、この項目が正規表現と一致することをチェックします。
- **file_option** フィールドを CAN_BE_NULL に設定すれば、ファイルが存在していない場合でも合格にできます。

- **file_option** フィールドを CAN_NOT_BE_NULL に設定すると、ファイルが存在して空の場合、不合格になります。

例:

```
<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

このチェックでは、リモート Windows システムが適切に機能するのにリモート レジストリ アクセスを必要とします。

使用法

```
<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>
```

このポリシー項目は、レジストリ キー (またはレジストリ項目) が存在しているかどうかをチェックします。このチェックは、関数 **RegOpenKeyEx** および **RegQueryValueEx** を呼び出すことにより実行されます。

許可されるタイプは次のとおりです。

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

key_item フィールドが指定されていない場合、キー パスの存在がチェックされます。このフィールドが指定されている場合は、キー項目の存在がチェックされます。

例:

```
<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

このチェックでは、リモート Windows システムが適切に機能するのにリモート レジストリ アクセスを必要とします。

使用法

```
<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>
```

このポリシー項目は、レジストリ キー値のチェックに使用されます。"セキュリティの設定 -> ローカル ポリシー -> セキュリティ オプション" 内の多くのポリシー チェックが、このポリシー項目を使用します。このチェックは、関数 **RegQueryValueEx** を呼び出すことにより実行されます。

reg_key フィールドは、レジストリ キーの名前を示します ("HKLM\SOFTWARE\Microsoft\Driver Signing" など)。キーの最初の部分 (HKLM) は、正しいレジストリ ハイブに接続するために使用されます。以降のパスは、目的の **reg_item** が保存されている静的な場所を示します。

HKU (HKEY_USERS) ハイブは特殊ケースになります。HKU キーに対して SID を指定することはできません。nbin が内部的に各 SID 上で繰り返され、各 SID の値が有効である場合のみ合格します。

例:

```
<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>
```

以下に対してループされます。

```
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
...
```

すべての SID に対して"ScreenSaveActive" が 1 に設定されている場合、合格になります。

オプションの **reg_option** フィールドを CAN_BE_NULL に設定すると、キーが存在していない場合でも強制的に合格にできます。または、CAN_NOT_BE_NULL に設定できます。

追加のオプション **reg_enum** を引数 "ENUM_SUBKEYS" で使用すると、レジストリ キーのすべてのサブキーに対する指定値を列挙できます。たとえば、キー **HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall** には、多くのソフトウェア パッケージが含まれています。"Uninstall" 以下のすべてのサブキーに対する "CurrentVersion" 値の照合を行いたい場合は、**reg_enum** を使用します。

例:

```
<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>
```

この HKU レジストリ ハイブの監査には、**reg_key** レジストリ パスに SID (セキュリティ識別子) が含まれていません。この例では、すべての HKU SID に対して指定の **reg_item** が検索されます。

例:

```
<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>
```

次のメイン **value_type** フィールド タイプを使用できます。

- POLICY_SET
  `value_data: "Enabled" or "Disabled"`
- POLICY_DWORD
  `value_data: DWORD or RANGE [same dword as in registry or range]`

- POLICY_TEXT
  value_data: "TEXT" [same text as in registry]

- POLICY_MULTI_TEXT
  value_data: "TEXT1" && "TEXT2" && ... && "TEXTN" [same texts as in registry]

- POLICY_BINARY
  value_data: "0102ac0b...34fb" [same binary as in registry]

- FILE_ACL、REG_ACL、SERVICE_ACL、LAUNCH_ACL、ACCESS_ACL
  value_data: "acl_name" [name of the acl to use]

次のオプション **value_type** フィールド タイプも使用できます。事前定義済み項目で使用されます。

- DRIVER_SET
  value_data: "Silent Succeed", "Warn but allow installation", "Do not allow installation"

- LDAP_SET
  value_data: "None" or "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" or "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" or "send ntlmv2 response only\refuse lm & ntlm"

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

- EVENT_METHOD
  value_data: "by days", "manually" or "as needed"
- POLICY_DAY
  value_data: DWORD or RANGE (time in days)
- POLICY_KBYTE
  value_data: DWORD or RANGE

custom_item フィールドに対しては、メイン **value_type** を使用します。オプション タイプは事前定義済み項目用に作成されています。

value_type が ACL の場合、レジストリ項目はバイナリ形式でセキュリティ記述子になっている必要があります。

例:

```
<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

このチェックでは、リモート Windows システムが適切に機能するのにリモート レジストリ アクセスを必要とします。

使用法

```
<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>
```

このポリシー項目は、レジストリ キーの ACL が正しいかどうかをチェックします。このチェックは、レジストリ キー ハンドルで関数 **RegGetKeySecurity** を呼び出すと実行されます。

許可されるタイプは次のとおりです。

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

reg_key フィールドに次の事前定義済みのパスを使用できます。

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

この監査を使用する際には、次の点に注意してください。

- **reg_key** フィールドには、ファイル レジストリ キーの完全パスを含める必要があります。
- **value_data** フィールドは、ポリシー ファイルで定義されている ACL の名前になります。
- **acl_option** フィールドは CAN_BE_NULL または CAN_NOT_BE_NULL に設定でき、キーが存在していない場合の合格/不合格の判断に使用されます。

例:

```
 <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>
```

上記のチェックが実行されると、コンプライアンス モジュールは、"**HKLM\SOFTWARE\Microsoft**" に対して定義されているパーミッションが registry_acl ACL2 に示されているパーミッションと一致しているかどうかをチェックします。

REGISTRY_AUDIT

このチェックでは、リモート Windows システムが適切に機能するのにリモート レジストリ アクセスを必要とします。

使用法

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

このポリシー項目は、レジストリ キーの ACL が正しいかどうかをチェックします。このチェックは、レジストリ キー ハンドルで関数 **RegGetKeySecurity** を呼び出すと実行されます。

許可されるタイプは次のとおりです。

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

reg_key フィールドに次の事前定義済みのパスを使用できます。

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

この監査を使用する際には、次の点に注意してください。

- **reg_key** フィールドには、ファイル レジストリ キーの完全パスを含める必要があります。
- **value_data** フィールドは、ポリシー ファイルで定義されている ACL の名前になります。

- **acl_option** フィールドは CAN_BE_NULL または CAN_NOT_BE_NULL に設定でき、キーが存在していない場合の合格/不合格の判断に使用されます。
- **acl_allow** フィールドと **acl_deny** フィールドは、"Successful" 監査イベントと "Failed" 監査イベントに対応します。

次のサンプル **.audit** ファイルでは、"HKLM\SOFTWARE\Microsoft" レジストリ キーの監査を、表示されていない "ACL2" という名前のアクセス制御リストに対して行います。

```
 <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

このチェックでは、リモート Windows システムが適切に機能するのにリモート レジストリ アクセスを必要とします。

使用法

```
<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>
```

このポリシー項目は、レジストリ キー タイプのチェックに使用されます。このチェックは、関数 **RegQueryValue** を呼び出すことにより実行されます。

reg_key フィールドは、レジストリ キー ("HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon") の名前になります。キーの最初の部分 (HKLM、HKU、HKCU...) は、正しいレジストリ ハイブに接続するために使用されます。ほとんどの場合、**reg_key** フィールドは、ワイルド カードを含まない静的なレジストリ エントリを必要としますが、HKU (HKEY_USERS) 内の値を検索する場合については例外が認められます。パスが HKU 以下にある場合、HKU 内のすべてのユーザー値に対して、指定パス下にある値が繰り返し検索されます。たとえば、reg_key: "HKU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" を **reg_item** "brastk" で指定した場合、HKU 以下の全ユーザーに対して、相対パス "HKU\<user_id>\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" 以下の "brastk" レジストリ キーの値が検索されます。たとえば、次のとおりです。

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

このチェックは、次のパス下で検索を行います。

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

オプション フィールド **reg_option** を CAN_BE_NULL に設定すると、キーが存在していない場合でも強制的に合格にできます。または、CAN_NOT_BE_NULL に設定できます。

このチェックで使用できる **value_type** は POLICY_TEXT のみです。

次のサンプル **.audit** ファイルでは、"HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon" のレジストリ タイプを監査します。

```
 <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>
```

HKCU の監査は、多くの Windows 上で機能しない可能性があります。HKCU の監査には "現在のユーザー" キーが必要ですが、Nessus 認証が SMB 上で行われる場合、通常、そのキーは存在しません。この問題を回避するためには、HKU (すべてのユーザー) の監査を行うことができます。プラグインが HKU キーの監査を検出すると、.DEFAULT キー以外、使用可能なすべての SID に対してループが自動的に発生します。このアプローチのデメリットは、システム ユーザー (SYSTEM、NT Authority など) も監査対象になるということです。この問題を回避するには、reg_ignore_hku_users を使用します。たとえば、次のとおりです。

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

これは、REGISTRY_SETTING チェックでのみ有効です。

SERVICE_PERMISSIONS

使用法

```
<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>
```

このポリシー項目は、サービス ACL が正しいかどうかをチェックします。このチェックは、サービス ハンドルで関数 **QueryServiceObjectSecurity** を呼び出すと実行されます。

許可されるタイプは次のとおりです。

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

この監査を使用する際には、次の点に注意してください。

- **value_data** フィールドは、ポリシー ファイルで定義されている ACL の名前になります。
- **acl_option** フィールドは CAN_BE_NULL または CAN_NOT_BE_NULL に設定でき、キーが存在していない場合の合格/不合格の判断に使用されます。

例:

```
<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>
```

上記のチェックが実行されると、コンプライアンス モジュールは、アラート サービスに対して定義されているパーミッションが service_acl ACL3 に示されているパーミッションと一致しているかどうかをチェックします。

SERVICE_AUDIT

使用法

```
<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>
```

このポリシー項目は、サービス ACL が正しいかどうかをチェックします。このチェックは、サービス ハンドルで関数 **QueryServiceObjectSecurity** を呼び出すと実行されます。

許可されるタイプは次のとおりです。

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

この監査タイプを使用する際には、次の点に注意してください。

- **value_data** フィールドは、ポリシー ファイルで定義されている ACL の名前になります。
- **acl_option** フィールドは CAN_BE_NULL または CAN_NOT_BE_NULL に設定でき、キーが存在していない場合の合格/不合格の判断に使用されます。
- **acl_allow** フィールドと **acl_deny** フィールドは、"Successful" 監査イベントと "Failed" 監査イベントに対応します。

次のサンプル **.audit** ファイルでは、"Alerter" サービスを監査します。

```
 <custom_item>
  type: SERVICE_AUDIT
  description: "Audit for Alerter Service"
  value type: SERVICE ACL
  value data: "ACL3"
  service: "Alerter"
 </custom_item>
```

WMI_POLICY

使用法

```
<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>
```

このチェックは、名前空間/クラス/属性内で指定される値を Windows WMI データベースで照会します。

使用されるシンタックスに応じて、キー値が抽出されるか、属性名が列挙されます。

許可されるタイプは次のとおりです。

```
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
```

システム上に重複値 ("MSFTPSVC/83207416" と "MSFTPSVC/2" など) を持つサービス構成から選択すると、リクエストによって、両方から選択された属性が抽出されます。片方がポリシー値と一致しない場合、**wmi_key** がレポートに追加され、どちらが一致しなかったかが示されます。**wmi_enum** フィールドを使用すると、比較またはポリシー値チェック対象の名前空間内の構成名を列挙できます。

デフォルトでは、WMI クエリが出力を戻さない場合、チェックはエラーをレポートします。この動作は変更可能です。**wmi_option** を CAN_BE_NULL に設定すると、チェックは合格となります。**only_show_query_output** を YES に設定すると、WMI クエリの出力が Nessus レポートに含まれます。**check_type** タグを使用すると、特定の文字列が出力に含まれていない限り、結果は合格になります。以下の例をご覧ください。

その他の考慮事項:

- WMI 属性は明示的に指定する必要があります。たとえば、`select * from foo` は機能しません。
- 値のない属性はレポートされません。
- 属性の大文字小文字は、Microsoft マニュアルで示されているとおりである必要があります。たとえば、属性 HandleCount は Handlecount であっても handlecount であっても無効になります。
- 配列タイプの値は結果に含まれません。

例 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>
```

システム上に 2 つの FTP サービス構成 ("MSFTPSVC/83207416" と "MSFTPSVC/2") がある場合、リクエストは、この両方から "UserIsolationMode" 属性を抽出します。片方がポリシー値 (0) と一致しない場合、**wmi_key** (この場合) がレポートに追加され、どちらが一致しなかったかが示されます。

例 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>
```

この例では、**value_data** で指定されている有効な 2 つの構成名があるかどうかがチェックされます。WMI 名前空間と関連属性について詳しく知りたい場合は、Microsoft の WMI CIM Studio を使用できます。この便利なツールは、以下のリンクからダウンロードできます。

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

例 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>
```

この例では、すべての Windows プロセスが一覧表示され、**svchost.exe** と **iPodService.exe** のインスタンスは削除されます。

項目

"項目" は、Windows コンプライアンス チェック エンジンで事前定義されているチェック タイプです。これらは、一般的な監査対象項目用に使用され、このため、監査チェック生成に必要なシンタックスが最小限になっています。"項目" の構造は次のとおりです。

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

name フィールドは、事前定義済みの名前である必要があります (事前定義済みの名前については、以下の「事前定義済みポリシー」の表を参照)。

事前定義済み項目はすべて、Windows 2003 SP1 上のドメイン ポリシー エディタで使用可能なリストに対応しています。

次の例では、最小パスワード長が 8 ～ 14 文字であるかどうかがチェックされます。

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

対応するカスタム項目は次のとおりです。

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

事前定義済みポリシー

ポリシー	使用法
パスワード ポリシー	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
アカウント ロックアウト ポリシー	name: "Account lockout duration" value: TIME_MINUTE or

	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 ポリシー	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
監査ポリシー	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
アカウント	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
監査	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	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
デバイス	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
ドメイン コントローラ	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

ドメイン メンバー	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
対話的ログオン	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 ネットワーク クライアント	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 ネットワーク サーバー	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
ネットワーク アクセス	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
ネットワーク セキュリティ	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
リカバリ コンソール	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
シャットダウン	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
システム暗号	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
システム オブジェクト	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
システム設定	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
イベント ログ	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

強制レポート

監査ポリシーは、"**report**" キーワードを使用して、特定の結果の出力を強制的に行うことができます。レポート タイプ PASSED、FAILED、および WARNING を使用できます。次に例を示します。

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

"**description**"フィールド内のテキストは、レポートにいつも表示されます。

このレポートは、Nessus により実行されるチェックが実際には完了しなかったことを監査担当者に通知したい場合に便利です。たとえば、特定のシステムの安全が物理的に確保されているかどうか判断するという要件があり、チェックまたは検査を手動で実行する監査担当者に伝えたい場合などに有効です。また、このタイプのレポートは、Nessus によって実行する必要がある特定の監査タイプが OVAL チェックで判断されなかった場合にも便利です。

条件

Windows ポリシーで、前提条件が有効である場合のみチェックを起動したり、複数のテストを 1 つにまとめるのに、**if/then/else** 論理を定義することができます。

条件を実行するシンタックスは次のとおりです。

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

条件タイプは "**and**" または "**or**" のいずれかになります。

上記の条件による監査では、項目 (またはカスタム項目) のリストになる "**then**" 文と "**else**" 文、つまり "**if**" 文が使用されています。"**else**" 文と "**then**" 文は、オプションで "**report**" タイプを使用して、条件の戻り値に応じて成功または失敗をレポートできます。

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

"**if**" 値が SUCCESS または FAILURE を戻します。この値は "**if**" 文が別の "**if**" 構造内にあるときに使用されます。たとえば、**<then>** 構造が実行されると、戻り値は、次のいずれかになります。

- 監査に項目のみ含まれる場合:すべての項目が成功した場合 SUCCESS が返され、それ以外の場合は FAILURE が返される。
- 監査に **<report>** のみ含まれる場合:レポート タイプが返される。
- 監査に項目と **<report>** の両方が含まれる場合:レポート タイプが返される。

<report> 文が使用されていて、タイプが "FAILED" の場合、失敗した理由と、重大度 (定義されている場合) がレポートに表示されます。

次に、パスワード ポリシーを監査する例を示します。"**and**" タイプが使用されるため、このポリシーが監査に通るには、両方のカスタム項目が成功する必要があります。この例では、非常に変わった有効なパスワード履歴ポリシーの組み合わせのためのテストとして、洗練されたテスト ロジックが実装されています。

```
<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>
```

上記の例では、新しい "**report**" タイプのみが表示されますが、**if/then/else** 構造は、"**else**" 句内の追加監査の実行をサポートします。1 つの条件内で、入れ子になった **if/then/else** 句を使用することもできます。より複雑な例を次に示します。

```
<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>
```

この例では、Administrator アカウントの名前が変更されていない場合、パスワード期間が 30 日以下であるかどうかの監査が行われます。この監査ポリシーは、パスワード ポリシーにかかわらず Administrator アカウントの名前が変更されていれば、合格となり、Administrator アカウントが変更されていない場合のみ、パスワード有効期間ポリシーのテストが発生します。

Windows コンテンツ監査コンプライアンス ファイル リファレンス

Windows コンテンツ **.audit** チェックは、システム構成設定を列挙するのではなく、機密データを含む特定のファイル タイプを Windows ファイル システムから検索することを目的とする点について、Windows 構成 **.audit** チェックと異なります。このチェックには、監査担当者が検索パスワードを絞り込み、非準拠データを効率的に識別、表示できる幅広いオプションが含まれます。

引用符の用法:

監査フィールドを囲む一重引用符と二重引用符は、次の 2 つの場合を除いてどちらでも使用可能です。

1. CRLF など、文字をリテラルに解釈する必要がある特殊フィールドを含む Windows コンプライアンス チェックでは、一重引用符を使用します。文字列として解釈する組み込みフィールドはエスケープする必要があります。

たとえば、次のとおりです。
```
expect: 'First line\r\nSecond line\r\nJohn\'s Line'
```

2. 二重引用符は WindowsFiles の "include_paths" と "exclude_paths" を使用する際に必要です。

どのフィールドタイプ (description、value_data、regex など) においても、一重引用符または二重引用符を含む文字列を使用する場合は、次のいずれかの方法を用います。

a. 外側の引用符とは異なる引用符タイプを使用する。

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

b. 組み込み引用符はバックスラッシュでエスケープする (二重引用符のみ)。

例:
```
expect: "\"Text to be searched\""
```

チェック タイプ

すべての Windows コンテンツ コンプライアンス チェックは、**check_type** カプセルで囲われ、"**WindowsFiles**" 指定を含んでいる必要があります。これは、他のすべての **.audit** ファイルと非常に似ています。コンテンツ チェック ファイルの基本フォーマットは次のとおりです。

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

各項目に対する実際のチェック内容は含まれていません。以降のセクションで、特定のコンテンツ項目監査の入力に使用できるさまざまなキーワードとパスワードについて説明します。

項目のフォーマット

使用法

```
<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>
```

これらの項目を使用して、さまざまなデータ型のさまざまなファイル フォーマットを監査できます。次の表に、サポートされているデータ型を示します。次のセクションでは、これらのキーワードを使用してさまざまなタイプのファイル コンテンツを監査できる多くの例を紹介します。

キーワード	説明
type	常に FILE_CONTENT_CHECK に設定する必要があります。
description	これは、SecurityCenter における一意のコンプライアンス脆弱性のタイトルとして使用される情報になります。Nessus によってレポートされる最初のデータ セットでもあります。
file_extension	Nessus によって検索されるすべての拡張子のリストになります。この拡張子は、"." なしで、引用符で囲み、パイプ記号で区切ります。**regex** や **expect** などのオプションが監査に含まれていなければ、指定の file_extension のファイルが監査出力に表示されます。
regex	このキーワードは、データの複雑なタイプの検索に使用される正規表現を格納します。正規表現が一致すると、最初に一致したコンテンツが脆弱性レポートに表示されます。 **regex** キーワードは以下の **expect** キーワードとともに実行する必要があります。 Windows コンプライアンス チェックとは異なり、Windows ファイル コンテンツ コンプライアンス チェックでは、**regex** と **expect** により、検索ファイル全体で同じデータ文字列の照合を行う必要はありません。Windows ファイル コンテンツ チェックでは、**regex** 文と **expect** 文の両方で、検索ファイルの <max_size> バイト内でのみデータの照合を行う必要があります。
expect	**expect** 文は、ドキュメント内で照合する 1 つ以上のシンプル パターンを列挙するのに使用します。たとえば、社会保障番号を検索する場合、"SSN"、"SS#"、または "Social" の検索が必要になります。 複数のパターンは、引用符で囲み、パイプ文字で区切ります。 このキーワードでは、ピリオド付きのシンプル パターン照合もサポートされます。文字列 "C.T" の照合では、**expect** 文は "CAT"、"CaT"、"COT"、"C T" などと一致します。 **expect** キーワードは、単一のパターン照合に対してはスタンドアロンで実行できますが、**regex** キーワードを使用する場合は **expect** が必須です。 Windows コンプライアンス チェックとは異なり、Windows ファイル コンテンツ コンプライアンス チェックでは、**regex** と **expect** により、検索ファイル全体で同じデータ文字列の照合を行う必要はありません。Windows ファイル コンテンツ チェックでは、**regex** 文と **expect** 文の両方で、検索ファイルの <max_size> バイト内でのみデータの照合を行う必要があります。

file_name	**file_extension** キーワードは必須ですが、このキーワードにより、分析対象のファイル リストを絞り込むことができます。パターン リストを提供することにより、ファイルの破棄または照合を実行できます。 たとえば、このキーワードにより、"employee"、"customer"、"salary" などの用語がファイル名に含まれている任意のタイプのファイル名を簡単に検索できます。
max_size	パフォーマンス上の理由で、各ファイルの最初の部分のみを監査対象とすることができます。このキーワードで監査対象バイトを指定できます。バイト数は引数として使用できます。また、キロバイトとして "K"、メガバイトとして "M" の拡張子を使用することもできます。
only_show	クレジット カード番号などの機密データの照合については、たとえば、レポートに表示される部分を最後の 4 桁のみにできます。このキーワードでは、ポリシーによって指定されるバイト分をレポートに表示できます。
regex_replace	このキーワードは、レポートに表示される正規表現内のパターンを制御します。クレジット カード番号など複雑なデータ パターンを検索する場合、最初に一致するのが目的のデータとは限りません。このキーワードは、より正確に目的のデータをキャプチャできる柔軟性を提供します。
include_paths	このキーワードにより、検索結果内にディレクトリまたはドライブを含めることができます。このキーワードは、"**exclude_paths**" キーワードと関連させて使用することも、また独立して使用することもできます。これは、特定のドライブまたはフォルダのみをマルチドライブ システム上で検索する場合に特に便利です。パスは二重引用符で括り、パスを複数列挙する場合は、パイプ記号で区切ります。  "**include_paths**" キーワードで指定できるのは、ドライブ文字またはフォルダ名のみです。ファイル名を "**include_paths**" 値文字列に含めることはできません。
exclude_paths	このキーワードにより、検索結果からドライブ、ディレクトリ、またはファイルを除外できます。このキーワードは、"**include_paths**" キーワードと関連させて使用することも、また独立して使用することもできます。これは、特定のドライブ、ディレクトリ、またはファイルを検索結果から除外する必要がある場合に特に便利です。パスは二重引用符で括り、パスを複数列挙する場合は、パイプ記号で区切ります。
see_also	このキーワードにより、参照リンクを含めることができます。 例: `see_also: "https://benchmarks.cisecurity.org/tools2/linux/CIS_Redhat_Linux_5_Benchmark_v2.0.0.pdf"`
solution	このキーワードにより、必要に応じて "ソリューション" テキストを含めることができます。 例: `solution : "Remove this file, if its not required"`
reference	このキーワードにより、**.audit** に相互参照を含めることができます。フォーマットは "ref|ref-id1,ref|ref-id2" です。 例:

	reference : "CAT|CAT II,800-53|IA-5,8500.2|IAIA-1,8500.2|IAIA-2,8500.2|IATS-1,8500.2|IATS-2"

コマンド ライン例

このセクションでは、**.tns** 拡張子を持つフェイク テキスト ドキュメントを作成し、このドキュメントに対して複数の簡単または複雑な **.audit** ファイルを実行します。サンプルの監査ファイルごとに、サポートされる Windows コンテンツ パラメータを見ていきます。

nasl コマンド ライン バイナリも使用します。ここで紹介する各 **.audit** ファイルは、Nessus 4 または SecurityCenter スキャン ポリシーに簡単にドロップできますが、1 つのシステムのクイック監査については、この方法が非常に効率的です。**/opt/nessus/bin** ディレクトリから毎回実行するコマンドは次のとおりです。

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

<IP> は、監査対象システムの IP アドレスです。

Nessus 4 では、**.nbin** (またはその他のプラグイン) を実行する際、ターゲット システムの資格情報と **.audit** ファイルの保存場所を入力する必要があります。

ターゲット テスト ファイル

使用するターゲット ファイルには、次のコンテンツが含まれます。

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

このデータを認証アクセス対応の Windows システムにコピーします。ファイル名を "Tenable_Content.tns" にします。

例 1:"Nessus" の単語が含まれる .tns ドキュメントを検索する

次に、ドキュメント内のどこかに "Nessus" の単語を含む **.tns** ファイルを検索するシンプルな **.audit** ファイルを示します。

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

このコマンドを実行すると、次の出力が生成されます。

```
"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
```

この結果は、一致があったことを示しています。検出されたデータが探していたデータではなかったため、レポートでは、"FAILED" と表示されています。たとえば、社会保障番号を探す監査を実行して、パブリック コンピュータ上で社会保障番号の一致があっても、コンプライアンス上の理由で不合格として記録されます。

例 2:"France" の単語が含まれる .tns ドキュメントを検索する

次に、ドキュメント内のどこかに "France" の単語を含む **.tns** ファイルを検索するシンプルな **.audit** ファイルを示します。

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

今回生成される出力は次のとおりです。

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

監査対象の **.tns** ファイルに "France" がまったく含まれていなかったので、この監査に合格しました。

例 3:"Nessus" の単語が含まれる .tns ドキュメントと .doc ドキュメントを検索する

Microsoft Word ドキュメントを検索するための 2 つ目の拡張子の追加は非常に簡単です。次の例をご覧ください。

```
<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>
```

テスト コンピュータ上に、次の結果が表示されます。

```
"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
```

テスト **.tns** ファイルは例 1 と同じ"不合格" ですが、2 つ目の **.doc** ファイルにも "Nessus" の単語が含まれています。このテストを独自のシステム上で実行した場合、"Nessus" の単語が含まれる Word ファイルが存在しているかどうかはそのシステムによります。

例 4:"Nessus" の単語と 11 桁の数字が含まれる .tns ドキュメントと .doc ドキュメントを検索する

11 桁の数字との照合を行う最初の正規表現を追加します。必要なことは、この正規表現を表す **regex** キーワードを、同じ **.audit** ファイルに追加するだけです。

```
<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>
```

次の出力が生成されます。

```
"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)
```

最後の例で一致した **.doc** ファイルは検索が続いています。11 桁の数字が含まれていないので、表示がストップしています。また、**regex** キーワードを使用しているので、一致内容がデータに表示されています。

これには、10 桁の数字の検出も含まれています。上記の 11 桁の数字には 2 つの 10 桁数字が含まれています (0123456789 と 1234567890)。11 桁ちょうどの一致を求める場合は、次のことを示す正規表現を追加します。

"他の数字が前後に来ない 11 桁の数字を検索する"

このことを正規表現で表すには、次のような "not" 演算子を使用します。

```
<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>
```

左から右に読むと、"^" 文字とドル記号が数回出てきます。"^" は、行の開始を意味する場合と、負の一致を表す場合があります。ドル記号は、行末を意味します。上記の正規表現は、数字で始まらないが改行で始まっていることは可能で、11 桁を含み、後ろに数字がないか、行末で終わっているパターンを探します。正規表現は行の開始と終了を特殊ケースとして扱うので、"^" 文字または "$" 文字の使用は必須です。

例 5:"Nessus" の単語と 11 桁の数字が含まれる .tns ドキュメントと .doc ドキュメントを検索し、最後の 4 バイトしか表示しない

.audit ファイルにキーワード **only_show** を追加すると、出力が制限されます。監査担当者による検索対象の機密データへのアクセスを制限できます。

```
<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>
```

一致データは、次のように "X" 文字で隠されます。

```
"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)
```

例 6:最初の 50 バイトに "Correlation" の単語が含まれる .tns ドキュメントを検索する

この例では、**max_size** キーワードを使用します。テスト ファイルでは、"Correlation" の単語は 50 バイト以上先にあります。

```
<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>
```

これを実行した結果は合格になります。

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

max_size 値を "50" から "50K" に変更して、スキャンを再実行します。これはエラーになります。

```
"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
```

例 7:出力表示を制御する

この例では、**regex_replace** キーワードを使用します。次の **.audit** ファイルを実行します。

```
<check_type:"WindowsFiles">
<item>
type: FILE_CONTENT_CHECK
description: "Seventh Example"
file_extension: "tns"
```

```
regex: "Passive Vulnerability Scanner"
expect: "Nessus"
</item>
</check_type>
```

次のような出力が生成されます。

```
"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)
```

ここでもし、"Passive" 部分と "Scanner" 部分の照合を行う正規表現が必要ではあるが、戻り値としては "Vulnerability" 部分だけがほしい場合は、次のような正規表現になります。

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

ただしここでも、全体一致の "Passive Vulnerability Scanner" が戻されます。これは、正規表現文は、最初の一致として文字列全体を処理するからです。2 つ目の項目を取得したい場合は、**regex_replace** キーワードを追加する必要があります。

```
<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>
```

スキャンにより次の出力が返されます。

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

"\3" を使用することにより、一致内の 2 つ目の項目が返されます。最初の一致である "\1" では、文字列全体が返されます。"\2" と指定すると "Passive" が返され、"\4" と指定すると "Scanner" が返されます。

このような機能がある理由ですが、クレジット カード番号など複雑なデータ パターンを検索する場合、最初に一致するのが目的のデータとは限りません。このキーワードにより、正確に目的のデータをキャプチャできる柔軟性が提供されます。

例 8:ファイル名をフィルタとして使用する

例 3 の **.audit** ファイルでは、**.tns** ファイルと **.doc** ファイル両方の結果が返されます。

```
<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>
```

テスト コンピュータ上に、次の結果が表示されます。

```
"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
```

file_name キーワードを使用すれば、必要なファイルまたは必要でないファイルをフィルタリングできます。このキーワードを **.audit** ファイルに追加し、名前に "tenable" が含まれているファイルのみ対象にする場合、次のようになります。

```
<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>
```

次のような出力が生成されます。

```
"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
```

.doc ファイルは、このパス内に "tenable" の単語が含まれていないので、出力に含まれていません。

一致文字列は正規表現なので、柔軟性が高く、必要なファイルと必要でないファイルを幅広く照合できます。たとえば、文字列 "[Tt]enable" と指定すると、"Tenable" または "tenable" が一致します。同様に、拡張子全体または拡張子の一部の照合を行いたい場合、"\.t" のようにバックスラッシュ付きのドットをエスケープする必要があります。このように指定すると、"t" で始まる任意の拡張子を検索できます。

例 9：inclusion/exclusion キーワードを使用する

"**include_paths**" キーワードと "**exclude_paths**" キーワードを使用して、ドライブ文字、ディレクトリ、およびファイル名の拡張子に基づいて、検索をフィルタリングできます。

```
<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>
```

次のような出力が生成されます。

```
Windows File Contents Compliance Checks
"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
```

この出力は、標準的な Windows ファイル コンテンツ検索結果とほぼ同じですが、除外パスが除外されています。"**include_paths**" を使用して 1 つのパス ("**c:**" など) を含めた場合、他のパスはすべて自動的に除外されます。また、ドライブ文字 ("**d:**" など) を除外して、そのドライブ以下のフォルダ ("**d:\users**" など) を含めた場合、"**exclude_paths**" キーワードの方が優先され、このドライブ全体が検索対象から外れます。ただし、ドライブ **c:** を含めて、その下のサブフォルダ (**C:\users**: など) を除外することはできます。

さまざまなタイプのファイル フォーマットの監査

どんなファイル拡張子でも監査対象にできますが、**.zip** や **.gz** などのファイルは実行中に解凍されません。圧縮ファイルまたはデータ内に何らかのエンコーディングが行われているファイルについては、パターン検索ができません。

Unicode フォーマットでデータが格納されているドキュメントに対しては、**.nbin** ファイルの解析ルーチンが、出現するすべての "NULL" バイトを削除します。

また、Microsoft Office ドキュメントについてはすべてのバージョンをサポートします。ここには、**.xlsx** や **.docx** などの Office 2007 で追加されたエンコーディングされたバージョンも含まれます。

さまざまなタイプの PDF ファイル フォーマットもサポートされます。Tenable は、照合用にロー文字列を抽出する拡張 PDF アナライザーを提供しています。ユーザー側で考えることは、PDF ファイルで検索するデータの種類のみです。

パフォーマンスについての考慮事項

デフォルトの **.audit** ファイルを変更して、ライブ ネットワーク上でテストを行う場合、考慮すべきいくつかのトレードオフがあります。

- 検索すべき拡張子はどれか。
- どれだけの量のデータをスキャンするのか。

.audit ファイルに **max_size** キーワードは必須ではありません。このキーワードがない場合、Nessus はファイル全体を取得しようとし、一致するパターンが見つかるまで、スキャンが続けられます。このようなファイルはネットワークをスキャンするので、通常のスキャンや構成監査よりも多くのネットワーク トラフィックが消費されます。

複数の Nessus スキャナが SecurityCenter によって管理されている場合、データは、スキャンされる Windows ホストから脆弱性監査を実行するスキャナに移動するだけで済みます。

Cisco IOS 構成監査コンプライアンス ファイル リファレンス

ここでは、Cisco IOS コンプライアンス チェックのフォーマットと関数、および各設定の論理的根拠について説明します。

引用符の用法:

監査フィールドを囲む一重引用符と二重引用符は、次の 2 つの場合を除いてどちらでも使用可能です。

1. CRLF など、文字をリテラルに解釈する必要がある特殊フィールドを含む Windows コンプライアンス チェックでは、一重引用符を使用します。文字列として解釈する組み込みフィールドはエスケープする必要があります。

たとえば、次のとおりです。

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

2. 二重引用符は WindowsFiles の "include_paths" と "exclude_paths" を使用する際に必要です。

どのフィールドタイプ (description、value_data、regex など) においても、一重引用符または二重引用符を含む文字列を使用する場合は、次のいずれかの方法を用います。

a. 外側の引用符とは異なる引用符タイプを使用する。

例:

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

b. 組み込み引用符はバックスラッシュでエスケープする (二重引用符のみ)。

例:

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

チェック タイプ

すべての Cisco IOS コンプライアンス チェックは、**check_type** カプセルで囲われ、"Cisco" 指定を含んでいる必要があります。これは、この **.audit** ファイルが、Cisco IOS オペレーティング システムを実行しているシステム用であることを識別するためのものです。

例:

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

他のコンプライアンス監査タイプとは異なり、追加のタイプやバージョンのキーワードはありません。

キーワード

次の表に、Cisco コンプライアンス チェックにおける各キーワードの使用方法を示します。

キーワード	使用例とサポートされている設定
type	CONFIG_CHECK、CONFIG_CHECK_NOT、RANDOMNESS_CHECK "CONFIG_CHECK" は、指定項目が CISCO IOS "show config" 出力に存在していることを確認します。また、"CONFIG_CHECK_NOT" は、指定項目が存在していないことを確認します。"RANDOMNESS_CHECK" は、文字列の複雑度のチェック (パスワード チェックなど) に使用されます。regex を使用して検索対象の項目が指定されている場合に、その文字列が "ランダム" (大文字小文字が混在し、数値と特殊文字が 1 つ以上含まれる 8 文字長以上) であることをチェックします。  ランダム度チェック パラメータは、現在では構成不可です。
description	"**description**" キーワードは、実行するチェックの簡単な説明を追加します。**description** フィールドには一意のテキストを指定して、**description** フィールドが重複するチェックは作成しないことを強く推奨します。Tenable の SecurityCenter は、一意のプラグイン ID 番号を **description** フィールドに基づいて自動的に生成します。 例: `description: "Forbid Remote Startup Configuration"`
feature_set	"**feature_set**" キーワードは Unix コンプライアンス チェックの "**system**" キーワードに似ています。Cisco IOS の Feature Set のバージョンをチェックして、結果生じるチェックを実行するか、regex の失敗によりチェックをスキップします。この機能は、チェックが特定 Feature Set のシステムにのみ適用可能な場合に有効です。 例: `<item>` `type: CONFIG_CHECK` `description: "Version Check"` `info: "SSH Access Control Check."` `feature set: "K8" context:"line .*"` `item:"access-class [0-9]+ in"` `</item>` 上記のチェックでは、Feature Set バージョンが特定の regex (K8) と一致する場合にのみ "item" チェックが実行されます。 Feature Set バージョン チェックで失敗すると、以下のようなエラーが表示されます。 `"Version Check" : [SKIPPED]` `Test defined for the feature set 'K8' whereas we are running C850-ADVSECURITYK9-M`
ios_version	"**ios_version**" キーワードは Unix コンプライアンス チェックの "**system**" キーワードに似ています。Cisco IOS のバージョンをチェックして、結果生じるチェックを実行するか、regex の失敗によりチェックをスキップします。この機能は、チェックが特定 IOS のシステムにのみ適用可能な場合に有効です。

	例: `<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>` 上記のチェックでは、IOS バージョンが特定の regex (12\.[5-9]) と一致する場合にのみ "item" チェックが実行されます。 IOS バージョン チェックで失敗すると、以下のようなエラーが表示されます。 `"Version Check" : [SKIPPED]` `Test defined for 12.[5-9] whereas we are running 12.4(15)T10`
info	"**info**" キーワードは、実行するチェックに関する詳細な説明を追加するのに使用します。チェックの論理的根拠として、規則であったり、詳細情報が記載されている URL であったり、企業ポリシーなどを指定できます。複数の **info** フィールドを、テキストをパラグラフとしてフォーマットするために 1 行に 1 つずつ追加することができます。使用できる **info** フィールド数に制限はありません。 各 "**info**" タグは、改行なしで別の行に入力する必要があります。複数行が必要な場合は (フォーマット上の理由などにより)、"**info**" タグを追加します。 例: `info: "Verify at least one local user exists and ensure"` `info: "all locally defined user passwords are protected"` `info: "by encryption."`
item	"**item**" キーワードは、監査対象の "show config" 出力内の構成項目を指定します。 例: `item:"transport input ssh"` このキーワードでは正規表現を使用して、照合結果をフィルタリングすることもできます。**regex** 機能の詳細については、**regex** キーワードの説明を参照してください。
regex	"**regex**" キーワードにより、構成項目設定の検索で、特定の正規表現への照合を行うことができます。 例: `regex: "snmp-server community ([^ ]*) .*"` 次のメタ文字には特別な処理が必要です:+ \ * ( ) ^ これらの文字は、リテラルで解釈する場合は、2 つのバックスラッシュでエスケープするか、角括弧 "[]" で括ります。. ? " ' などの他の文字は、リテラルで解釈するには、円記号を 1 つだけ使用します。

	これは、コンパイラがこれらの文字を処理する方法に関係します。
min_occurrences	"**min_occurrences**" キーワードは、監査に通るために必要な構成項目の最小出現回数を指定します。 例: `min_occurrences:"3"`
max_occurrences	"**max_occurrences**" キーワードは、監査に通るために必要な構成項目の最大出現回数を指定します。 例: `max_occurrences: "1"`
required	"**required**" キーワードは、監査対象項目がリモート システム上に存在する必要があるかないかを指定するのに使用します。たとえば、**required** が "NO" で、チェック タイプが "CONFIG_CHECK" の場合、構成項目が存在していてもしていなくても監査に通ります。**required** が "YES" の場合は、上記のチェックは不合格になる場合があります。 例: `required: NO`
context	"**context**" キーワードは、特定の構成項目の複数のインスタンスが存在している場合に便利です。たとえば、次のような構成があったとします。 `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` 特定のシリアル回線の値をテストしたい場合、"line" 値の **item** キーワードを使用するだけでは、複数の "line" オプションがあるため不十分です。このような場合、"**context**" を使用すると、対象項目を絞り込むことができます。次に例を示します。 `context: "con 0"` 次の構成項目でのみ grep 検索することができます。 `line con 0` ` no modem enable` このキーワードでは正規表現を使用して、照合結果をフィルタリングすることもできます。**regex** 機能の詳細については、**regex** キーワードの説明を参照してください。

コマンド ライン例

ここでは、Cisco IOS コンプライアンス チェック用に使用される一般的な監査例をいくつか紹介します。**nasl** コマンド ライン バイナリは、実行中に簡単に監査をテストするための方法として使用されます。以下の各 **.audit** ファイルは、Nessus スキャンポリシ

一に簡単にドロップできます。1 つのシステムのクイック監査については、コマンド ライン テストが効率的です。**/opt/nessus/bin** ディレクトリから毎回実行するコマンドは次のとおりです。

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

<IP> は、監査対象システムの IP アドレスです。

"enable" パスワードは必須です。

```
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 :
```

正しい "enable" ログイン パラメータについては、Cisco 管理者に問い合わせてください。

例 1:定義済み SNMP ACL を検索する

定義済みの "deny" SNMP ACL を検索するシンプルな **.audit** ファイルを次に示します。検出されない場合は、不合格メッセージが表示されます。このチェックは、ルータ IOS バージョンが指定の正規表現と一致した場合のみ実行されます。それ以外の場合は、チェックはスキップされます。

```
<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>
```

このコマンドを実行すると、準拠システムであれば、次の出力が生成されます。

```
"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.
```

不合格監査では、次の出力が返されます。

```
"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
```

この場合、"deny ip" ルールを検索したが検出されなかったので、チェックが不合格になりました。

例 2:"finger" サービスが無効になっていることを確認する

リモート ルータ上で安全でない "**finger**" サービスを検索するシンプルな **.audit** ファイルを次に示します。このチェックは、ルータ IOS バージョンが指定の正規表現と一致した場合のみ実行されます。それ以外の場合は、チェックはスキップされます。サービスが検出された場合、不合格メッセージが表示されます。

```
<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>
```

このコマンドを実行すると、準拠システムであれば、次の出力が生成されます。

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

Disable finger server.
```

不合格監査では、次の出力が返されます。

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

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

例 3:SNMP コミュニティ文字列とアクセス制御のランダム度を検証するランダム度チェック

ランダム度が足りない SNMP コミュニティ文字列を検索するシンプルな **.audit** ファイルを次に示します。ランダム度が足りないと判断されるコミュニティ文字列が検出されると、不合格メッセージが表示されます。"required" オプションが "NO" に設定されているので、SNMP コミュニティ文字列が存在していない場合でも合格になります。このチェックは、ルータが Feature Set "K9" を使用している場合のみ実行されます。それ以外の場合は、チェックはスキップされます。

```
<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>
```

このコマンドを実行すると、準拠システムであれば、次の出力が生成されます。

```
"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.
```

不合格監査では、次の出力が返されます。

```
"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
```

上記の場合、十分にランダムなトークンを持たない 2 つの文字列 "foobar" と "public" が検出されたため、チェックは不合格になっています。

例 4：SSH アクセス制御を検証するコンテキスト チェック

"**context**" キーワードを使用してすべての "line" 構成項目を検索し、**regex** を実行して SSH アクセス制御が設定されているかどうかをチェックするシンプルな **.audit** ファイルを次に示します。

```
<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>
```

このコマンドを実行すると、準拠システムであれば、次の出力が生成されます。

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

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

不合格監査では、次の出力が返されます。

```
"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
```

上記の例では、"**context**" キーワードの regex "**line .***" と一致する 2 つの文字列がありました。どの line にも "**item**" regex が含まれていないので、"FAILED" メッセージが返されています。

条件

Cisco 監査ポリシー内に **if/then/else** 論理を定義することができます。これにより、監査が合格した場合、合格/不合格の結果ではなく、警告メッセージを返すことができます。

条件を実行するシンタックスは次のとおりです。

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

例:

```
<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>
```

"サイレント" チェックなので、条件の失敗または成功はレポートに表示されません。

条件タイプは "**and**" または "**or**" のいずれかになります。

Juniper 構成監査コンプライアンス ファイル リファレンス

ここでは、Juniper コンプライアンス チェックのフォーマットと関数、および各設定の論理的根拠について説明します。

引用符の用法:

監査フィールドを囲む一重引用符と二重引用符は、次の 2 つの場合を除いてどちらでも使用可能です。

1. CRLF など、文字をリテラルに解釈する必要がある特殊フィールドを含む Windows コンプライアンス チェックでは、一重引用符を使用します。文字列として解釈する組み込みフィールドはエスケープする必要があります。

たとえば、次のとおりです。

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

2. 二重引用符は WindowsFiles の "include_paths" と "exclude_paths" を使用する際に必要です。

どのフィールドタイプ (description、value_data、regex など) においても、一重引用符または二重引用符を含む文字列を使用する場合は、次のいずれかの方法を用います。

a. 外側の引用符とは異なる引用符タイプを使用する。

例:

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

b. 組み込み引用符はバックスラッシュでエスケープする (二重引用符のみ)。

例:

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

チェック タイプ:CONFIG_CHECK

Juniper コンプライアンス チェックは、**custom_item** カプセルで囲われ、CONFIG_CHECK または SHOW_CONFIG_CHECK のいずれかになります。これらは、他の **.audit** ファイルと同じように処理され、Juniper オペレーティング システム (Junos) を実行するシステムに対して使用されます。CONFIG_CHECK チェックは、2 つ以上のキーワードで構成されます。キーワード **type** と **description** は必須です。この後に 1 つ以上のキーワードが続きます。チェックは、"set" フォーマットで構成を監査することにより実行されます。

"set" フォーマットの構成は、"display set" を "show configuration" リクエストに追加することにより取得できます。たとえば、次のとおりです。

```
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"
```

キーワード

次の表に、Juniper コンプライアンス チェックにおける各キーワードの使用方法を示します。

キーワード	使用例とサポートされている設定
type	CHECK_CONFIG と SHOW_CHECK_CONFIG "CHECK_CONFIG" は、"set" フォーマットの Juniper "show configuration" 出力に指定の構成項目が存在していることを確認します。同様に、"SHOW_CONFIG_CHECK" は、デフォルト フォーマットの "show configuration" 出力に構成項目が存在していることを確認します。
description	"**description**" キーワードは、実行するチェックの簡単な説明を追加します。**description** フィールドには一意のテキストを指定して、**description** フィールドが重複するチェックは作成しないことを強く推奨します。Tenable の SecurityCenter は、一意のプラグイン ID 番号を **description** フィールドに基づいて自動的に生成します。 例: `description: "3.1 Disable Unused Interfaces"`
info	"**info**" キーワードは、実行するチェックに関する詳細な説明を追加するのに使用します。チェックの論理的根拠として、規則であったり、詳細情報が記載されている URL であったり、企業ポリシーなどを指定できます。複数の **info** フィールドを、テキストをパラグラフとしてフォーマットするために 1 行に 1 つずつ追加することができます。使用できる **info** フィールド数に制限はありません。  各 "**info**" タグは、改行なしで別の行に入力する必要があります。複数行が必要な場合は (フォーマット上の理由などにより)、"**info**" タグを追加します。 例: `info: "Review the the list of interfaces"`

	info: "Disable unused interfaces"
severity	"**severity**" キーワードは、実行するチェックの重大度を指定します。 例: severity: MEDIUM 重大度は HIGH、MEDIUM、または LOW に設定できます。
regex	"**regex**" キーワードにより、構成項目設定の検索で、特定の正規表現への照合を行うことができます。 例: regex: "set system syslog .+" 次のメタ文字には特別な処理が必要です:+ \ * ( ) ^ これらの文字は、リテラルで解釈する場合は、2 つのバックスラッシュでエスケープするか、角括弧 "[]" で括ります。. ? " ' などの他の文字は、リテラルで解釈するには、バックスラッシュを 1 つだけ使用します。 これは、コンパイラがこれらの文字を処理する方法に関係します。 チェックに "**regex**" タグが設定され、"**expect**"、"**not_expect**"、"**number_of_lines**" タグは設定されていない場合、チェックは、regex と一致する全行をレポートします。
expect	このキーワードにより、"**regex**" タグに一致する構成項目を監査できます。また、"**regex**" タグが使用されていない場合、構成全体から "**expect**" 文字列が検索されます。 例: expect: "syslog host 1.1.1.1" "**regex**" によって検出された構成行が "**expect**" タグと一致している場合、チェックは合格になります。"**regex**" が設定されていない場合は、"**expect**" 文字列が構成から検出された場合に合格になります。 例: regex: "syslog host [0-9\.]+" expect: "syslog host 1.1.1.1" 上記の場合は、"**expect**" タグにより、syslog ホストが 1.1.1.1 に設定されていることがチェックされます。
not_expect	このキーワードにより、構成内に含まれてはならない構成項目を検索できます。 例: not_expect: "syslog host 1.1.1.1" これは "**expect**" の反対の動作になります。"**regex**" によって検出された構成行が "**not_expect**" タグと一致しない場合、チェックは合格になります。"**regex**" タグが設定されていない場合は、構成に "**not_expect**" 文字列が検出されなければ合格です。 例: regex: "syslog host [0-9\.]+" not_expect: "syslog host 1.1.1.1"

	上記の場合は、"**not_expect**" タグにより、syslog ホストが 1.1.1.1 に設定されていないことがチェックされます。
number_of_lines	このキーワードにより、構成によって返される一致行数に基づいて監査チェックのコンプライアンスをテストできます。 `<custom_item>` `type: CONFIG_CHECK` `description: "Syslog"` `regex: "syslog host [0-9\.]+"` `number_of_lines: "^1$"` `</custom_item>` 上記の例では、"**regex**" と一致した 1 行のみ返される場合、チェック合格になります。

CONFIG_CHECK の例

Juniper デバイスに対する CONFIG_CHECK の使用例を次に示します。

```
<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>
```

チェック タイプ：SHOW_CONFIG_CHECK

このチェックは、多くの点で、CONFIG_CHECK .audit チェックによって監査される設定と同じ設定を監査します。ただし、監査対象構成のフォーマットが異なります。SHOW_CONFIG_CHECK は、デフォルト フォーマットの構成を監査します。

たとえば、デフォルト フォーマットの構成例を次に示します。

```
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;
}
```

CONFIG_CHECK より高い柔軟性が必要でない限り、このチェックは推奨されません。SHOW_CONFIG_CHECK では、各 SHOW_CONFIG_CHECK .audit チェックが個別にコマンドを Juniper デバイス上で実行するため、CPU オーバーヘッドが多く、完了までの時間が長くなります。このチェックは、監査担当者に柔軟性を提供し、CONFIG_CHECK では効率的に監査できないような将来の事例をサポートします。

キーワード

次の表に、Junos コンプライアンス チェックにおける各キーワードの使用方法を示します。チェックのコンプライアンスは、"**expect**"、"**not_expect**"、または"**number_of_lines**"タグのいずれかとチェック出力を比較することにより判断されます。コンプライアンス テスト タグは複数指定できません ("**expect**" と "**not_expect**" が重複しない状態で "**expect**"、"**not_expect**"、または "**number_of_lines**" のいずれかを指定できます)。

キーワード	使用例とサポートされている設定
hierarchy	このキーワードにより、Junos 構成内の特定の階層に移動できます。 例: `hierarchy: "interfaces"` hierarchy キーワードは、SHOW_CONFIG_CHECK では "show configuration" コマンドに内部的に追加されます。たとえば、次のとおりです。 `<custom_item>` ` type: SHOW_CONFIG_CHECK` ` description: "3.6 Forbid Multiple Loopback Addresses"` ` hierarchy: "interfaces"` `</custom_item>` 上記のチェックは次のコマンドを実行するのと同じです。 `show configuration interfaces`
property	このキーワードにより、Junos デバイス上の特定の "**property**" を監査できます。デフォルトでは、SHOW_CONFIG_CHECK は、**match**、**except**、**find** といった 1 つ以上のキーワードに続く "show configuration" コマンドを監査します。"**property**" キーワードが設定さ

	れている場合、特定のプロパティが監査されます。 例: `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> ``` 上記のチェックは次のコマンドを実行するのと同じです。 ``` show ospf interface detail ``` 上記の例は、他の例のように "show configuration" を実行しません。
find	このキーワードは、SHOW_CONFIG_CHECK **.audit** チェックで適切な構成階層を検出します。 ``` find: "chap" ``` **find** キーワードは、"show configuration" リクエストに追加されます。 ``` <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> ``` 上記のチェックは次のコマンドを実行するのと同じです。 ``` show configuration interfaces | find "chap" | match "access- profile" ```
match	このキーワードは、SHOW_CONFIG_CHECK .audit チェックの一致行を検索します。 ``` match: "multihop" ``` **match** キーワードは、"show configuration" リクエストに追加されます。

	<custom_item> type: SHOW_CONFIG_CHECK description: "3.6 Forbid Multiple Loopback Addresses" hierarchy: "interfaces" match: "lo[0-9]" </custom_item> 上記のチェックは次のコマンドを実行するのと同じです。 show configuration interfaces | match "lo[0-9]"
except	このキーワードは、SHOW_CONFIG_CHECK .audit チェックで構成から特定行を除外します。 except: "multihop" **except** キーワードは、"show configuration" リクエストに追加されます。 <custom_item> type: SHOW_CONFIG_CHECK description: "6.8.1 Require External Time Sources" hierarchy: "system ntp" match: "server" except: "boot-server" </custom_item> 上記のチェックは次のコマンドを実行するのと同じです。 show configuration system ntp | match "server" | except "boot-server"
expect	このキーワードにより、"**regex**" タグに一致する構成項目を監査できます。また、"**regex**" タグが使用されていない場合、構成全体から "**expect**" 文字列が検索されます。"**regex**" によって検出された構成行が "**expect**" タグと一致している場合、チェックは合格になります。"**regex**" が設定されていない場合は、"**expect**" 文字列が構成から検出された場合に合格になります。 regex: "syslog host [0-9\.]+" expect: "syslog host 1.2.4.5" 上記の場合、"**expect**" タグは、複雑度が 1 ～ 4 の値に設定されていることを確認します。 expect: "syslog host" 上記の場合、"**expect**" タグは、複雑度が 4 に設定されていることを確認します。

not_expect	このキーワードにより、構成内に含まれてはならない構成項目を検索できます。 これは "**expect**" の反対の動作になります。"**regex**" によって検出された構成行が "**not_expect**" タグと一致しない場合、チェックは合格になります。"**regex**" タグが設定されていない場合は、構成に "**not_expect**" 文字列が検出されなければ合格です。 `regex: "syslog host [0-9\.]+"` `not_expect: "syslog host 1.2.3.4"` `not_expect: "syslog host"`
number_of_lines	このキーワードにより、構成によって返される一致行数に基づいて監査チェックのコンプライアンスをテストできます。 `<custom_item>` `type: CONFIG_CHECK` `description: "Syslog"` `regex: "syslog host [0-9\.]+"` `number_of_lines: "^1$"` `</custom_item>` 上記の例では、"**regex**" と一致した 1 行のみ返される場合、チェック合格になります。

SHOW_CONFIG_CHECK の例

Juniper デバイスに対する SHOW_CONFIG_CHECK の使用例を次に示します。

```
<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>
```

条件

Juniper 監査ポリシー内に **if/then/else** 論理を定義することができます。条件により、1 つのファイルで複数の構成を処理できるようになります。

条件を実行するシンタックスは次のとおりです。

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

例:

```
<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>
```

条件はレポート内には表示されません。つまり、条件が失敗したか成功したかは表示されません ("サイレント" チェック)。

条件タイプは "**and**" または "**or**" のいずれかになります。

レポート

<then> または <else> 内で実行して、目的の PASSED/FAILED 条件を適用できます。

```
<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>
```

"report type" の有効値は PASSED、WARNING、および FAILED です。

Check Point GAiA 構成監査コンプライアンス ファイル リファレンス

ここでは、Check Point GAiA コンプライアンス チェックのフォーマットと関数、および各設定の論理的根拠について説明します。

引用符の用法:

監査フィールドを囲む一重引用符と二重引用符は、次の 2 つの場合を除いてどちらでも使用可能です。

1. CRLF など、文字をリテラルに解釈する必要がある特殊フィールドを含む Windows コンプライアンス チェックでは、一重引用符を使用します。文字列として解釈する組み込みフィールドはエスケープする必要があります。

たとえば、次のとおりです。

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

2. 二重引用符は WindowsFiles の "include_paths" と "exclude_paths" を使用する際に必要です。

どのフィールドタイプ (description、value_data、regex など) においても、一重引用符または二重引用符を含む文字列を使用する場合は、次のいずれかの方法を用います。

a. 外側の引用符とは異なる引用符タイプを使用する。

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

b. 組み込み引用符はバックスラッシュでエスケープする (二重引用符のみ)。

例:
```
expect: "\"Text to be searched\""
```

チェック タイプ:CONFIG_CHECK

Check Point コンプライアンス チェックは、**custom_item** カプセルで囲まれ、CONFIG_CHECK が指定されているものになります。これは、他の **.audit** ファイルと同じように処理され、Check Point GAiA オペレーティング システムを実行するシステムに対して使用されます。CONFIG_CHECK チェックは、2 つ以上のキーワードで構成されます。キーワード **type** と **description** は必須です。この後に 1 つ以上のキーワードが続きます。チェックは、デフォルトで "set" フォーマットの "**show config**" コマンド出力を監査することより行われます。

キーワード

次の表に、GAiA コンプライアンス チェックにおける各キーワードの使用方法を示します。

キーワード	使用例とサポートされている設定
type	"CHECK_CONFIG" は、GAiA "show configuration" 出力に指定の構成項目が存在していることを確認します。
description	"**description**" キーワードは、実行するチェックの簡単な説明を追加します。**description** フィールドには一意のテキストを指定して、**description** フィールドが重複するチェックは作成しないことを強く推奨します。Tenable の SecurityCenter は、一意のプラグイン ID 番号を **description** フィールドに基づいて自動的に生成します。 例: `description: "1.0 Require strong Password Controls - 'min-password-length >= 8'"`
info	"**info**" キーワードは、実行するチェックに関する詳細な説明を追加するのに使用します。チェックの論理的根拠として、規則であったり、詳細情報が記載されている URL であったり、企業ポリシーなどを指定できます。複数の **info** フィールドを、テキストをパラグラフとしてフォーマットするために 1 行に 1 つずつ追加することができます。使用できる **info** フィールド数に制限はありません。  各 "**info**" タグは、改行なしで別の行に入力する必要があります。複数行が必要な場合は (フォーマット上の理由などにより)、"**info**" タグを追加します。 例: `info: "Enable palindrome-check on passwords"`

severity	"**severity**" キーワードは、実行するチェックの重大度を指定します。 例: `severity: MEDIUM` 重大度は HIGH、MEDIUM、または LOW に設定できます。
regex	"**regex**" キーワードにより、構成項目設定の検索で、特定の正規表現への照合を行うことができます。 例: `regex: "set snmp .+"` 次のメタ文字には特別な処理が必要です:+ \ * ( ) ^ これらの文字は、リテラルで解釈する場合は、2 つのバックスラッシュでエスケープするか、角括弧 "[]" で括ります。. ? " ' などの他の文字は、リテラルで解釈するには、バックスラッシュを 1 つだけ使用します。 これは、コンパイラがこれらの文字を処理する方法に関係します。 チェックに "regex" タグが設定され、"expect"、"not_expect"、"number_of_lines" タグは設定されていない場合、チェックは、regex と一致する全行をレポートします。
expect	このキーワードにより、"**regex**" タグに一致する構成項目を監査できます。また、"**regex**" タグが使用されていない場合、構成全体から "**expect**" 文字列が検索されます。 "**regex**" によって検出された構成行が "**expect**" タグと一致している場合、チェックは合格になります。"**regex**" が設定されていない場合は、"**expect**" 文字列が構成から検出された場合、合格になります。 例: `regex: "set password-controls complexity"` `expect: "set password-controls complexity [1-4]"` 上記の場合、"**expect**" タグは、複雑度が 1 ～ 4 の値に設定されていることを確認します。
not_expect	このキーワードにより、構成内に含まれてはならない構成項目を検索できます。 これは "**expect**" の反対の動作になります。"**regex**" によって検出された構成行が "**not_expect**" タグと一致しない場合、チェックは合格になります。"**regex**" タグが設定されていない場合は、構成に "**not_expect**" 文字列が検出されなければ合格です。 例: `regex: "set password-controls password-expiration"` `not_expect: "set password-controls password-expiration never"` 上記の場合は、"**not_expect**" タグにより、password-controls が "never" に設定されていないことがチェックされます。

CONFIG_CHECK の例

Check Point デバイスに対する CONFIG_CHECK の使用例を次に示します。

```
<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>
```

条件

Check Point 監査ポリシー内に **if/then/else** 論理を定義することができます。条件により、1 つのファイルで複数の構成を処理できるようになります。

条件を実行するシンタックスは次のとおりです。

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

例:

```
<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>
```

条件はレポート内には表示されません。つまり、条件が失敗したか成功したかは表示されません ("サイレント" チェック)。

条件タイプは "**and**" または "**or**" のいずれかになります。

レポート

<then> または <else> 内で実行して、目的の PASSED/FAILED 条件を適用できます。

```
<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>
```

"report type" の有効値は PASSED、WARNING、および FAILED です。

Palo Alto ファイアウォール構成監査コンプライアンス ファイル リファレンス

Palo Alto 用のコンプライアンス チェックは、他のコンプライアンス監査とは異なります。主な違いは、この監査では `XSL Transforms` (XSLT) を頻繁に使用して関連情報を抽出するという点です (詳細については、付録 C を参照してください)。ほとんどの API リクエストに対する Palo Alto ファイアウォールのレスポンスは XML フォーマットなので、監査用には XSLT が最も効率的な方法になります。XSLT に慣れていないユーザーでも、XSLT は、必要なデータを必要なフォーマットで抽出するための XML ファイルへのクエリ手法の 1 つ、と考えることができます。簡単に言えば、XSLT は、データベースに対する SQL のような働きをします。

Palo Alto 監査は、AUDIT_XML と AUDIT_REPORTS という 2 つのチェック タイプをサポートします。

AUDIT_XML

次に、Palo Alto AUDIT_XML のサンプル チェックを示します。

```
<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>
```

この監査には次の 4 つの基本要素が含まれています。

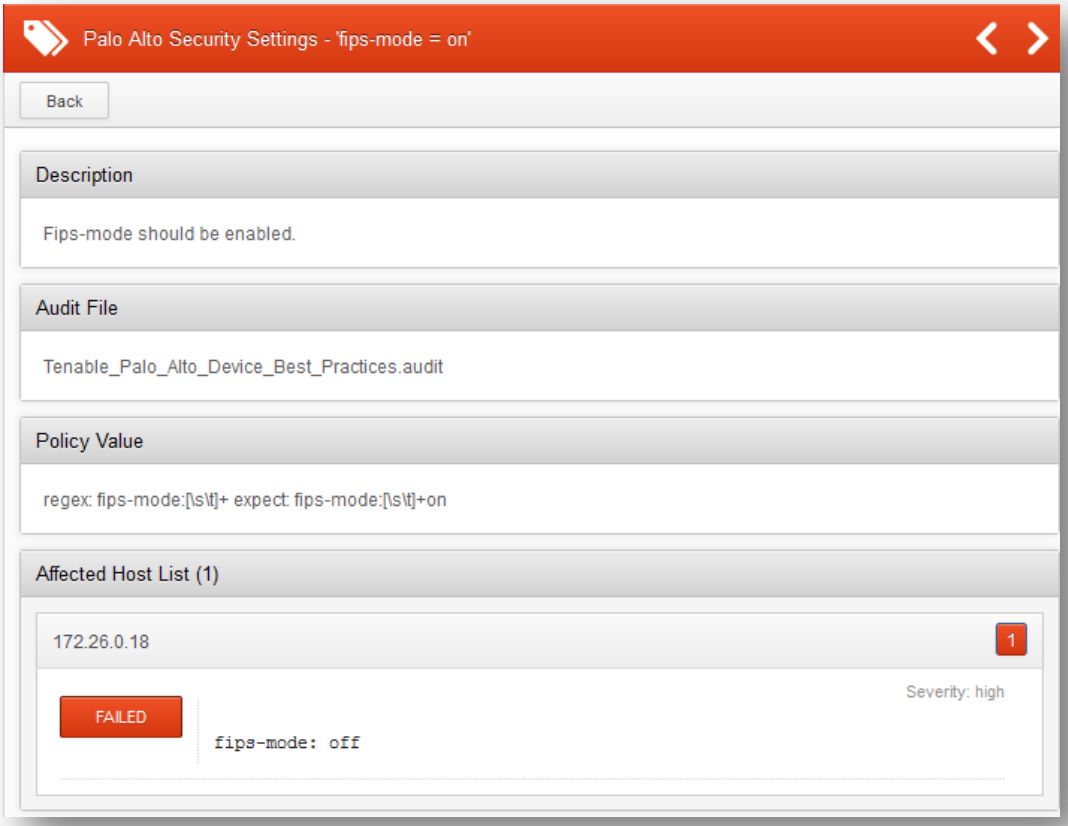
1. **`type`** は、監査のタイプ (ここでは XML の監査) を示し、**`description`** は監査を説明します。**`info`** キーワードは、レポートに関連テキストを含める方法を提供します。

2. **`api_request_type`** は、リクエストのタイプ (op == operational config) を示し、このリクエストは、最終的に実行される実際のリクエストです。現在、これがサポートされている唯一のリクエスト タイプです。

3. **`xsl_stmt`** キーワードは、API リクエストの実行後に返される XML に適用される XSLT を定義します。

4. 最後に、**`regex`** キーワードと **`expect`** キーワードは、コンプライアンス/構成監査を行うのに使用します。

上記のサンプル チェックにより、Nessus 上に次のレポートが生成されます。



AUDIT_REPORTS

Palo Alto ファイアウォールの優れた機能の 1 つは、継続的にネットワークがプロファイルされ、毎日 40 以上の事前定義済みのレポートが生成されることです。レポートは Top Applications、Top Attackers、Spyware Infected Hosts などです。また、管理者は、適宜、ダイナミック レポート (1 時間前など) を生成できます。Nessus は、これらのレポートに対して直接クエリを実行して、これを Nessus レポートに含めることができます。

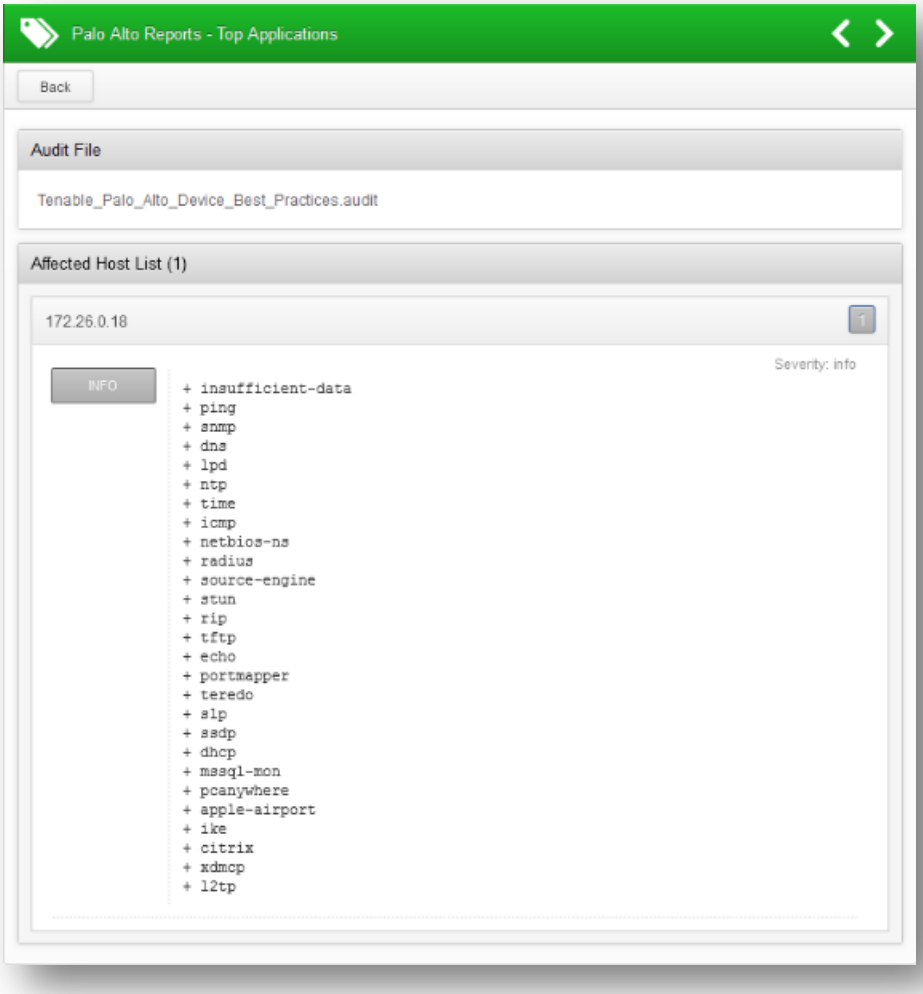
この機能には 2 つの利点があります。まず、ユーザーが同じデータを取得するのに別々のインタフェースをスキャンする必要がありません。次に、レポートの監査機能を提供できます。たとえば、ネットワーク内で使用するアプリケーションから Facebook を除外したい場合、Facebook が Top Applications レポートに含まれると、管理者は不合格レポートを生成できます。たとえば、次のとおりです。

```
<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>
```

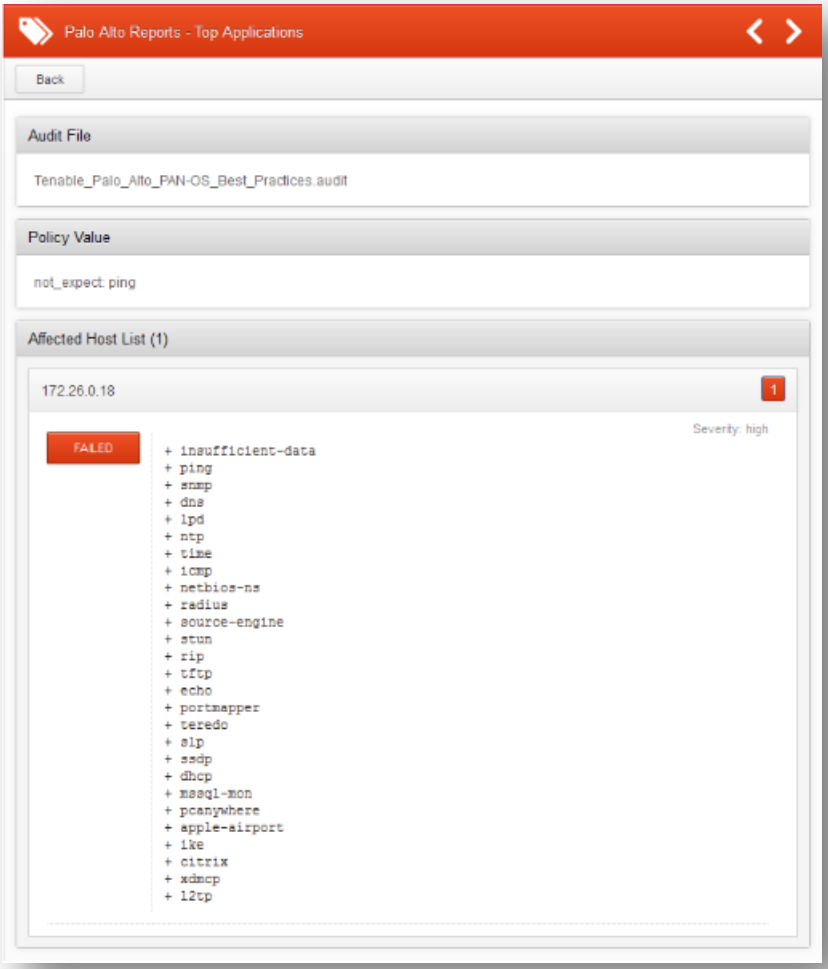
このレポートで **not_expect** キーワードを使用すると次のようになります。

```
<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>
```

最初の例では、次のようなレポートが返されます。



2 つ目の例では、不合格レポートが返されます。



キーワード

Palo Alto 監査でサポートされているキーワードは次のとおりです。

キーワード	説明
`type`	これは必ず AUDIT_XML または AUDIT_REPORTS に設定する必要があります。
`description`	これは、SecurityCenter における一意のコンプライアンス脆弱性のタイトルとして使用される情報になります。Nessus によってレポートされる最初のデータ セットでもあります。
`info`	このキーワードは、実行するチェックに関する詳細な説明を追加するのに使用します。指定できる `info` フィールド数に制限はありません。`info` コンテンツは、二重引用符で囲む必要があります。
`api_request_type`	このキーワードは、リクエストのタイプを示します。Palo Alto API は、6 つのタイプのリクエストをサポートしています。keygen、op、commit、reports、export、および config です。このプラグインの目的上、表示されるのは `op` リクエスト タイプのみです。

request	このキーワードは、ファイアウォール上で実行されるリクエストを指定します。各リクエストの結果がキャッチされるので、後続のリクエストの結果、別のリクエストが生じることはありません。加えて、AUDIT_REPORTS チェックの場合、デフォルトの Tenable 監査には 9 つのチェックのみ含まれます。さらに多くのレポートを含める場合には、新しいチェックを作成して、`type=report` の後に request キーワードを REST API URL で置換する必要があります。たとえば、次のとおりです。 `/api/?type=report&reporttype=predefined&reportname=hruser-top-url-categories`
regex	このキーワードは、特定の正規表現と一致する項目を検索します。チェックに **regex** キーワードが含まれ、**expect** も **not_expect** も設定されていない場合、チェックは、regex と一致するすべての行をそのままレポートします。

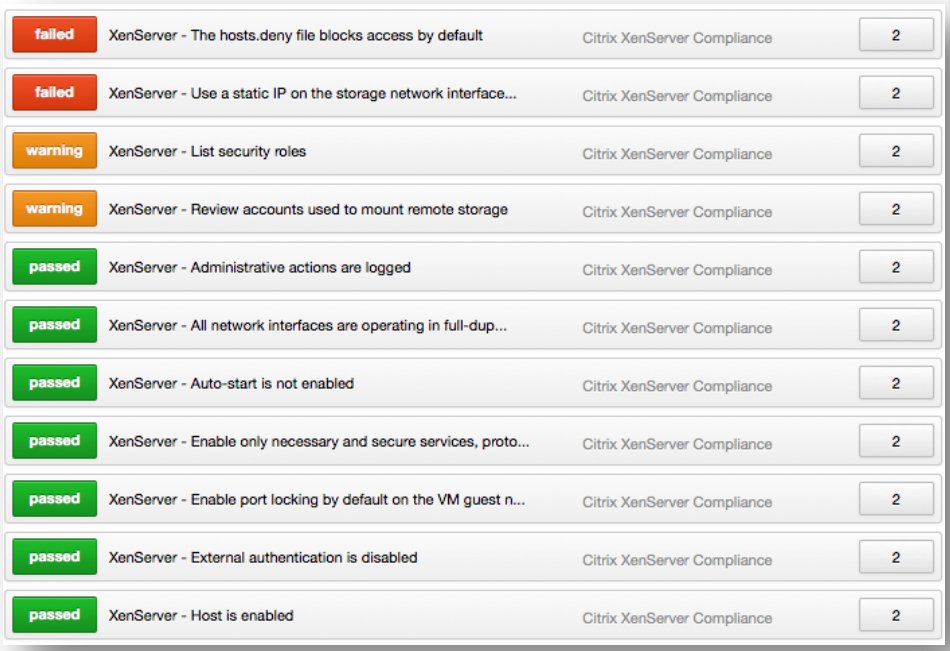
チェックのコンプラインスは、**expect** キーワードまたは **not_expect** キーワードのいずれかと、チェック出力との比較により判断されます。コンプライアンス テスト タグは複数指定できません (**expect** と **not_expect** のいずれか一方を指定できますが、**expect** と **not_expect** の両方を指定することはできません)。

キーワード	説明
expect	このキーワードにより、**regex** キーワードによって一致する構成項目を監査できます。また、**regex** キーワードが使用されていない場合、構成全体から **expect** 文字列が検索されます。**regex** によって検出された構成行が **expect** 文字列と一致している場合、チェックは合格になります。**regex** が設定されていない場合は、**expect** 文字列が構成から検出された場合のみ合格になります。
not_expect	このキーワードにより、構成内に含まれては**ならない**構成項目を検索できます。これは **expect** の反対の動作になります。**regex** によって検出された構成行が **not_expect** 文字列と一致しない場合、チェックは合格になります。**regex** キーワードが設定されていない場合は、構成に **not_expect** 文字列が検出されなければ合格です。

Citrix XenServer 監査コンプライアンス ファイル リファレンス

Citrix XenServer のコンプライアンス チェックは、1 点を除いて、以降説明する Unix 構成監査コンプライアンス ファイル リファレンス と非常に似ています。AUDIT_XE という追加監査を使用して、パッチ監査を実行できます。XenServer 監査で使用できるチェックタイプは次のとおりです。

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



チェック タイプ：AUDIT_XE

次に、XenServer AUDIT_XE のサンプル チェックを示します。

```
<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>
```

キーワード

次の表に、Citrix XenServer コンプライアンス チェックにおける各キーワードの使用方法を示します。

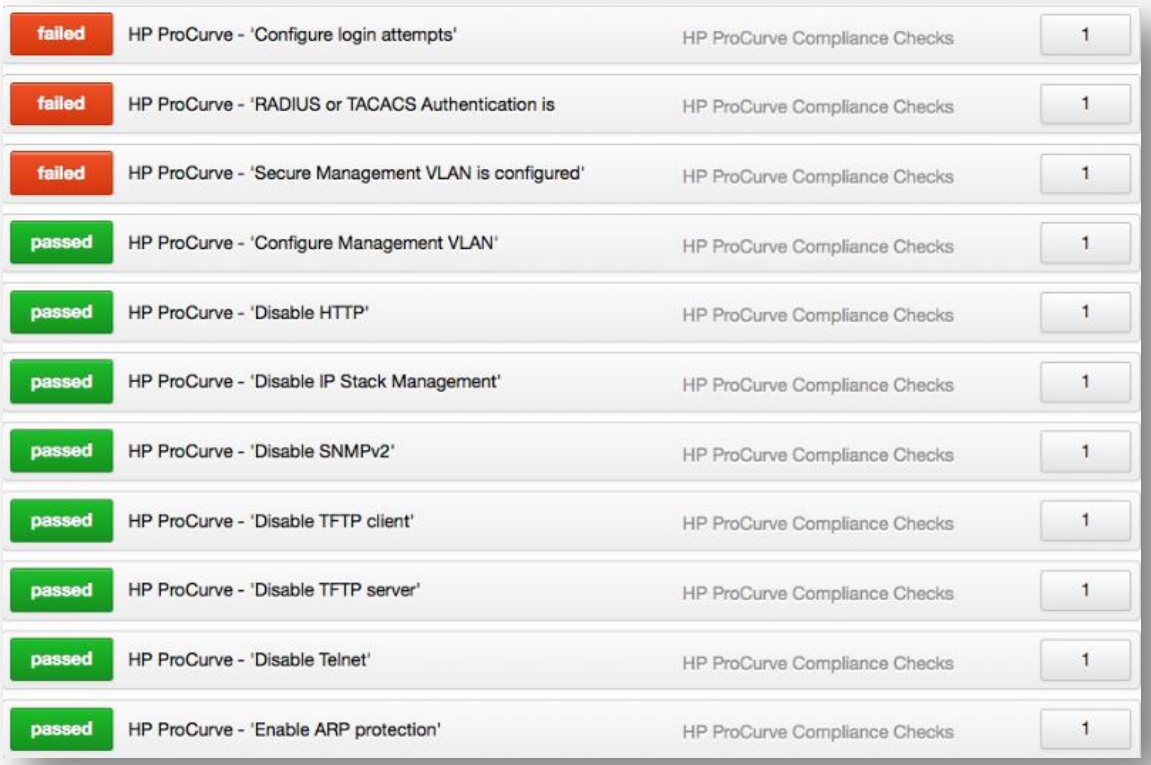
キーワード	使用例とサポートされている設定
type	AUDIT_XE
description	このキーワードは、実行するチェックを簡単に説明するのに使用します。description フィールドは一意である必要があり、2 つのチェックが同じ description フィールドを持つことはできません。これは、SecurityCenter が description フィールドを使用して、プラグイン ID 番号を自動生成するためです。 例： description: "List running VMs"

info	このキーワードは、実行するチェックに関する詳細な説明を追加するのに使用します。指定できる info フィールド数に制限はありません。info コンテンツは、二重引用符で囲む必要があります。 例: `info: "The allocated virtual CPUs (VCPU) should be reviewed. Desired settings depend on workload and operating system type."`
see_also	このキーワードにより、チェックについて役立つ情報を提供するリンクを含めることができます。 例: `see_also: "http://support.citrix.com/article/CTX137828"`
reference	このキーワードにより、監査チェックの相互参照を含めることができます。 例: `reference: "PCI|2.2.3,SANS-CSC|1"`
solution	このキーワードには、コンプライアンス エラーを修正するためのソリューション テキストを含めることができます。
severity	このキーワードは、チェックの重大度を設定するのに使用します。重大度は、HIGH、MEDIUM、または LOW のいずれかになります。 例: `severity: MEDIUM`
cmd	このキーワードは、ターゲット上で実行する **xe** コマンドを指定します。 例: `cmd: "/usr/bin/xe subject-list params=all"`
regex	このキーワードは、特定の正規表現と一致する項目を列挙します。"**regex**" キーワードが設定され、"**expect**" も "**not_expect**" も設定されていない場合、regex に一致するすべての項目がそのままレポートされます。 例: `regex: "power-state.+"`
expect	expect キーワードが指定されている場合、すべての結果が "**expect**" キーワードに一致する場合のみチェックは合格になります。1 つの結果でも **expect** キーワードと一致しない場合、すべての結果が **expect** と一致しないものとしてチェックは不合格になります。 例: `<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 .+"` ` expect: "running"` `</custom_item>`

not_expect	not_expect キーワードが設定されている場合、すべての結果が not_expect regex と一致しない場合のみチェックは合格になります。 例: `<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>`
ignore	このキーワードにより、結果から特定の項目を無視することができます。 例: `<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>`

HP ProCurve 監査コンプライアンス ファイル リファレンス

HP ProCurve 監査は、多くの点において、Cisco コンプライアンス プラグインの拡張と言えます。Tenable HP ProCurve 監査ファイルは、ProCurve スイッチ強化についての HP ホワイトペーパーに基づいています。この監査には、セキュリティ保護されていないサービスを無効にし、アクセス制御 (TACACS, RADIUS など) を有効にするためのチェックが含まれます。. 完全権限を持つ root または管理者の有効な SSH 資格情報が必要です。



チェック タイプ

HP ProCurve コンプライアンス チェックは、3 つのチェック タイプのうちの 1 つを使用します。監査用の一般的なシンタックスは次のとおりです。

```
<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>
```

キーワード

次の表に、HP ProCurve コンプライアンス チェックにおける各キーワードの使用方法を示します。

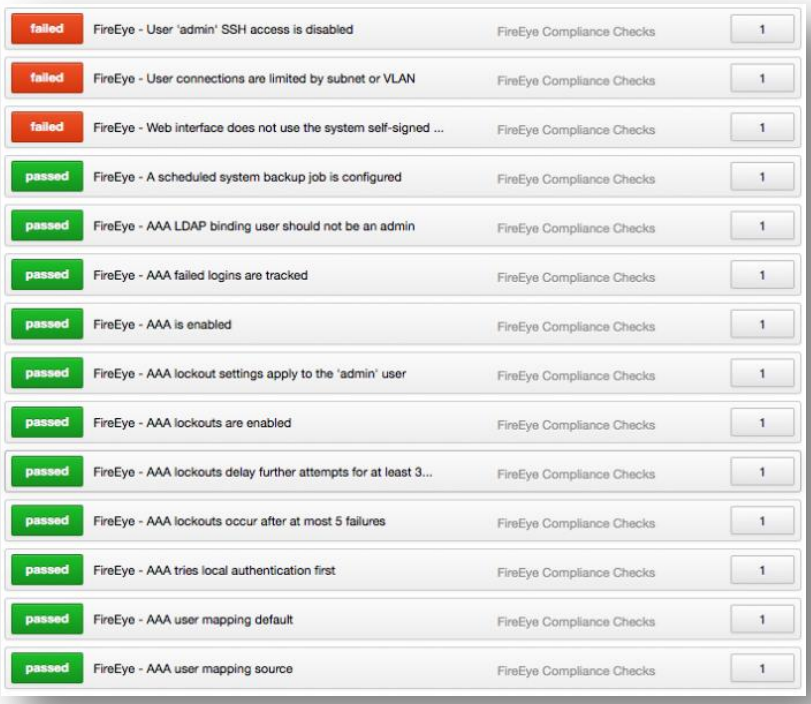
キーワード	使用例とサポートされている設定
type	CONFIG_CHECK CONFIG_CHECK_NOT RANDOMNESS_CHECK
description	このキーワードは、実行するチェックを簡単に説明するのに使用します。description フィールドは一意である必要があり、2 つのチェックが同じ description フィールドを持つことはできません。これは、SecurityCenter が description フィールドを使用して、プラグイン ID 番号を自

	動生成するためです。 例: `description: "Verify login authentication"`
info	このキーワードは、実行するチェックに関する詳細な説明を追加するのに使用します。指定できる info フィールド数に制限はありません。info コンテンツは、二重引用符で囲む必要があります。 例: `info: "Verifies login authentication configuration."`
see_also	このキーワードにより、チェックについて役立つ情報を提供するリンクを含めることができます。 例: `see_also: "http://www.hp.com/rnd/support/faqs/1800.htm"`
reference	このキーワードにより、監査チェックの相互参照を含めることができます。 例: `reference: "PCI|2.2.3,SANS-CSC|1"`
solution	このキーワードには、コンプライアンス エラーを修正するためのソリューション テキストを含めることができます。 例: `solution: "Modify the configuration to add missing line"`
severity	このキーワードは、チェックの重大度を設定するのに使用します。重大度は、HIGH、MEDIUM、または LOW のいずれかになります。 例: `severity: MEDIUM`
regex	このキーワードは、特定の正規表現と一致する項目を列挙します。"**regex**" キーワードが設定され、"**expect**" も "**not_expect**" も設定されていない場合、regex に一致するすべての項目がそのままレポートされます。 例: `regex: "power-state.+"`
item	このキーワードは、regex によって検出された行内での検索を行うためのものです。regex が設定されていない場合は、全行に対してチェックが行われます。 例: `regex: "power"`
context	このキーワードにより、特定のコンテキストに対して検索を行うことができます。コンテキストは、左端の行と、空白スペースを接頭辞にした任意の行で定義されます。 例: `context: "line .*"` コンテキストを利用して監査できるサンプル構成項目を次に示します。

	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> 上記のチェックでは、コンテキスト "vlan 1" に対して "ip address dhcp-bootp" が設定されていることが確認されます。
min_occurrences	このキーワードは、チェックの最小出現回数を設定できます。 例: min_occurrences: 3
max_occurrences	**min_occurrences** に似ていますが、ここでは最大出現回数を設定します。
required	このキーワードは、チェックの一致が必須かどうかを指定します。required フィールドの値は YES、NO、ENABLED、または DISABLED になります。 例: required: YES

FireEye 構成監査コンプライアンス ファイル リファレンス

FireEye 監査は、FireEye 用マニュアルと、一般的な基準ガイドラインに基づいています。この監査には、監査、識別、認証、アプライアンス管理、インテリジェント プラットフォーム管理インタフェース (IPMI)、有効なサービス、暗号化、およびマルウェア検出システム構成用のチェックが含まれます。完全権限を持つ root または管理者の有効な SSH 資格情報が必要です。



チェック タイプ

FireEye コンプライアンス チェックは、3 つのチェック タイプのうちの 1 つを使用します。監査用の一般的なシンタックスは次のとおりです。

```
<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>
```

キーワード

次の表に、HP ProCurve コンプライアンス チェックにおける各キーワードの使用方法を示します。

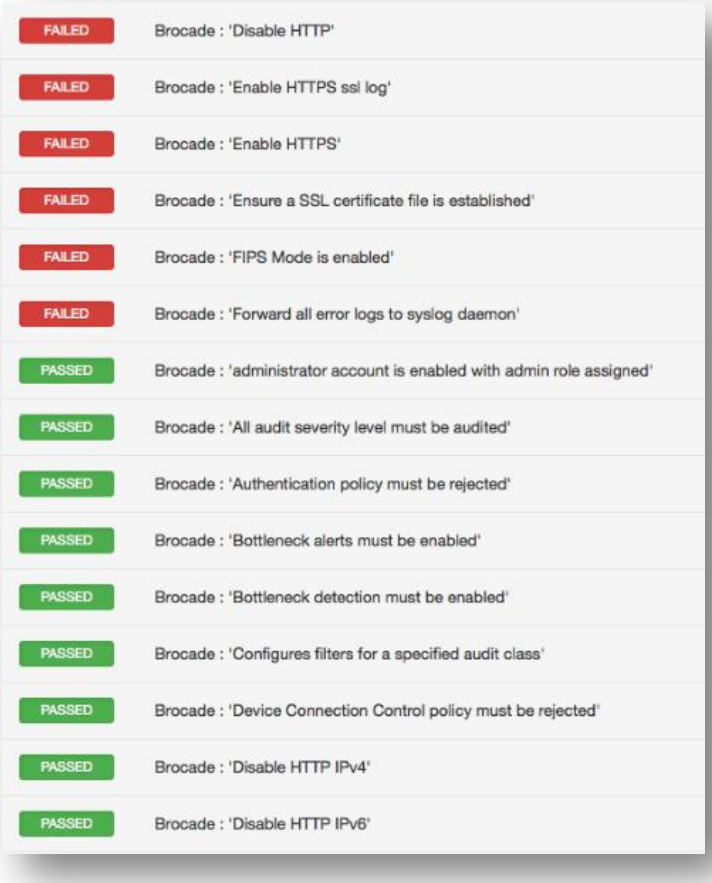
キーワード	使用例とサポートされている設定
type	CONFIG_CHECK CONFIG_CHECK_NOT RANDOMNESS CHECK
description	このキーワードは、実行するチェックを簡単に説明するのに使用します。description フィールドは一意である必要があり、2 つのチェックが同じ description フィールドを持つことはできません。これは、SecurityCenter が description フィールドを使用して、プラグイン ID 番号を自動生成するためです。 例: description: "Verify login authentication"

info	このキーワードは、実行するチェックに関する詳細な説明を追加するのに使用します。指定できる info フィールド数に制限はありません。info コンテンツは、二重引用符で囲む必要があります。 例: `info: "Verifies login authentication configuration."`
see_also	このキーワードにより、チェックについて役立つ情報を提供するリンクを含めることができます。 例: `see_also: "http://www.fireeye.com/support/"`
reference	このキーワードにより、監査チェックの相互参照を含めることができます。 例: `reference: "PCI|2.2.3,SANS-CSC|1"`
solution	このキーワードには、コンプライアンス エラーを修正するためのソリューション テキストを含めることができます。 例: `solution: "Modify the configuration to add missing line"`
severity	このキーワードは、チェックの重大度を設定するのに使用します。重大度は、HIGH、MEDIUM、または LOW のいずれかになります。 例: `severity: MEDIUM`
regex	このキーワードは、特定の正規表現と一致する項目を列挙します。"**regex**" キーワードが設定され、"**expect**" も "**not_expect**" も設定されていない場合、regex に一致するすべての項目がそのままレポートされます。 例: `regex: "power-state.+"`
expect	このキーワードは、regex によって検出された行内での検索を行うためのものです。regex によって検出されたすべての行が expect 設定と一致していれば合格です。regex が指定されていない場合は、すべての行がチェックされますが、検出する必要があるのは 1 行のみです。 例: `regex: "power"`
not_expect	**expect** と似ていますが、一致が検出された場合、不合格になります。**expect** と **not_expect** の両方とも指定されていない場合、適用可能なすべての行が info メッセージとしてレポートされます。
min_occurrences	このキーワードは、チェックの最小出現回数を設定できます。 例: `min_occurrences: 3`
max_occurrences	**min_occurrences** に似ていますが、ここでは最大出現回数を設定します。
required	このキーワードは、チェックの一致が必須かどうかを指定します。required フィールドの値は YES、NO、ENABLED、または DISABLED になります。 例:

	required: YES
cmd	これにより、show コマンドを実行できます。 例: cmd: "show version" "**show**" コマンドのみ指定可能です。 `<item>` `type: CONFIG_CHECK` `cmd: "show version"` `description: "Show Product version"` `regex: "Proaduct model:"` `expect: "1234"` `</item>`

BrocadeFabric OS (FOS) コンプライアンス ファイル リファレンス

Brocade Fabric OS (FOS) は、Fibre Channel スイッチおよび FICON スイッチの Brocade ファミリー上で実行されます。この監査には、パスワード ポリシー、有効サービス、ロックアウト ポリシー、セキュリティ保護されていないサービス構成、認証関連設定、ログ設定と監査設定に対するチェックが含まれます。完全権限を持つ root または管理者の有効な SSH 資格情報が必要です。



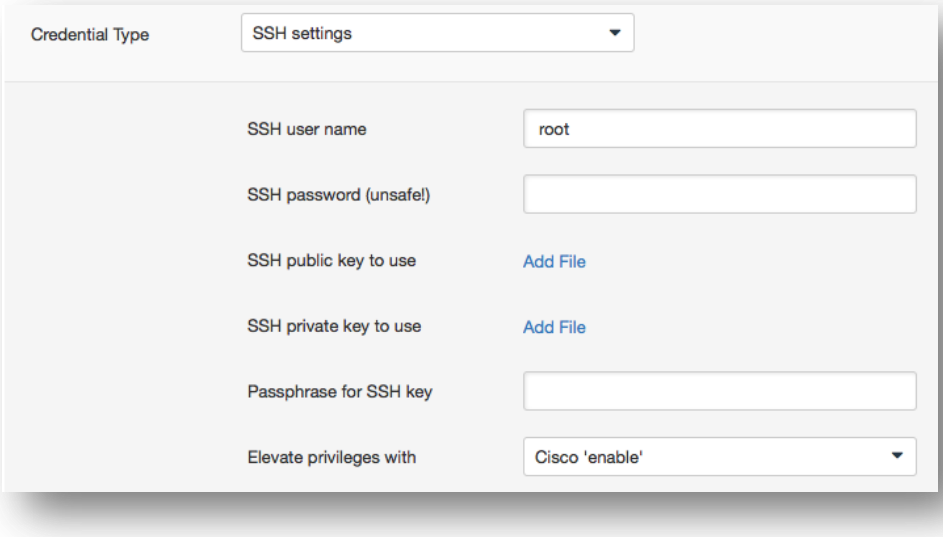
シンタックス

このプラグインと監査用のシンタックスは次のとおりです。

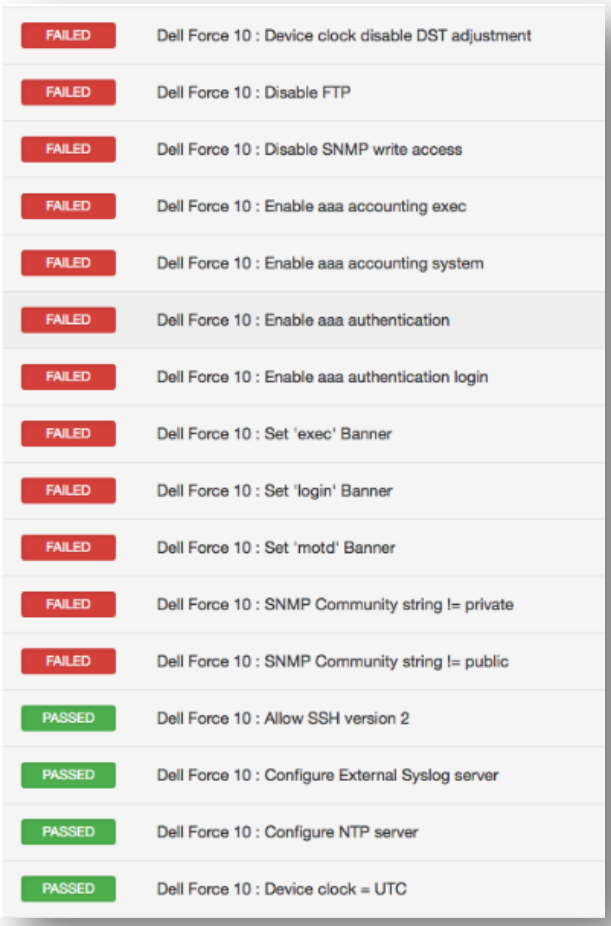
```
<custom_item>
 description: "Brocade : 'Enable SSH IPv4'"
 info: "SSH uses asymmetric authentication to exchange keys and create a secure
       encrypted session."
 info: "It is recommended that you use Secure Shell (SSH) instead of Telnet."
 see_also:
       "http://www.brocade.com/downloads/documents/product_manuals/B_SAN/FOS_CmdRef_v7
       00.pdf"
 solution: "The command to enable SSH is as follows\n
 switch:admin> ipfilter --addrule policy_name -rule rule_number -sip any -dp 22 -
       proto\n

tcp -act permit\n"

 reference: "SANS-CSC|11,SANS-CSC|10,PCI|2.2.3,800-53|CM-7,800-53|AC-1,800-53|SC-7"
 cmd: "ipfilter --show"
 context: "ipv4.+active"
 regex: "tcp\\s+22"
 expect: "permit"
</custom_item>
```

Dell Force10 コンプライアンス ファイル リファレンス

Dell Force10 (FTOS) デバイスは、さまざまな高機能スイッチで構成されます。この監査には、パスワード ポリシー、有効サービス、ロックアウト ポリシー、セキュリティ保護されていないサービス構成、認証関連設定、SNMP & NTP 構成、ログ設定と監査設定に対するチェックが含まれます。完全権限を持つ root または管理者の有効な SSH 資格情報が必要です。デバイス構成は "enable" モードからのみアクセスできます。



プリファレンスには、"cisco enable" という 1 つの "enable" オプションしかありません。このプラグインは、そのプリファレンスをピギーバックして、enable パスワードを設定します。したがって、ユーザーは、"cisco enable" オプションを使用して、enable パスワードを保存する必要があります。



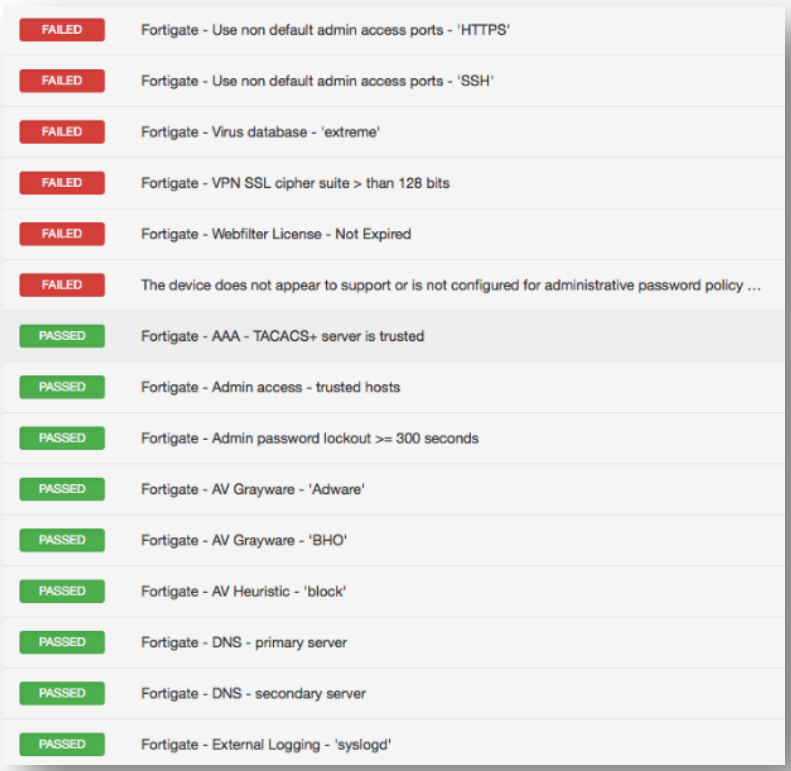
シンタックス

このプラグインと監査用のシンタックスは次のとおりです。

```
<custom_item>
 description: "Dell Force 10 : Min Password Length >= 8"
 info: "Passwords should be at least 8 characters in length"
 expect: "password-attributes.+min-length ([8-9]|[1-9][0-9]+)"
 solution: "To configure password length run the following command :\n

 password-attributes min-length 8"
 reference: "SANS-CSC|10,HIPAA|164.308(a)(5)(ii)(D),PCI|2.2.4,PCI|8.2.3"
</custom_item>
```

Fortinet FortiOS 監査コンプライアンス ファイル リファレンス

Fortinet FortiOS 監査には、パスワード ポリシー、マルウェア検出構成、有効サービス、ライセンスの情報とステータス、ログしきい値構成、NTP 構成、SNMP 構成、管理者ユーザー列挙、パッチ更新方法、監査構成とログ構成、および認証に対するチェックが含まれます。完全権限を持つ root または管理者の有効な SSH 資格情報が必要です。



シンタックス

このプラグインと監査用のシンタックスは次のとおりです。

```
<custom_item>
 description: "Fortigate - SSH login grace time <= 30 seconds"
 info: "SSH login grace time <= 30 seconds."
 reference: "HIPAA|HIPAA 164.308(a)(5)(ii)(D),SANS-CSC|16,PCI|2.2.3,800-53|AC-2(5)"
solution: "Issue the following command to configure SSH login grace time.

config system global
set admin-ssh-grace-time <time int>
end"
 context: "config system global"
 regex: "set[\\s]+admin-ssh-grace-time"
 expect: "set[\\s]+admin-ssh-grace-time[\\s]+([1-2][0-9]|30)$"
</custom_item>
```

description、**info**、**reference**、**solution** の各キーワードは、任意のテキストを含むことができ、目的はその名前のとおりです。これらのキーワードにより、**.audit** ファイル内のチェック関連のメタデータを含めることができます。**description** キーワードのみ必須で、その他のキーワードはすべてオプションです。

この監査は、設定が **regex**、**expect**、**not_expect** の各キーワードに準拠しているかいないかを検出します。Fortigate プラグインのリリース時点 (2014 年 1 月 21 日) では、Tenable は、コンプライアンス監査を実行するのに、これらのキーワードの 6 つの組み合わせをサポートする予定になっています。

regex、expect、not_expect なし

regex、**expect**、**not_expect** のどのキーワードも設定されていない場合、構成全体がレポートされます (**cmd** が指定されている場合は、コマンド出力全体がレポートされます)。

```
<custom_item>
 description: "Fortigate - HTTPS/SSH admin access strong ciphers"
 context: "config system global"
</custom_item>
```

上記のチェックでは、"config system global" コンテキスト全体がレポートされます。

regex のみ

regex のみ指定されている場合、regex に一致するすべての行がレポートされます。

```
<custom_item>
 description: "Fortigate - Review Admin Settings"
 context: "config system global"
 regex: "set[\\s]+admin-.+"
</custom_item>
```

これは情報提供のオプションです。たとえば、グローバル コンテキスト以下のすべての管理者設定の一覧が表示されます。一致する行がなかった場合、WARNING 結果が生成されます。ただし、**required** が YES に設定されている場合、チェックは不合格になります。

expect のみ

expect キーワードのみ指定されている場合、一致する行/構成項目が検出されると、チェックは合格になります。

```
<custom_item>
 description: "Fortigate - Admin password lockout = 300 seconds"
 context: "config system global"
 expect: "set[\\s]+admin-lockout-duration[\\s]+300$"
</custom_item>
```

上記のチェックでは、admin パスワード ロックアウトが 300 秒に設定されている場合、合格になります。

not_expect のみ

not_expect キーワードのみ指定されている場合、一致する行/構成項目が存在していなければ、チェックは合格になります。

```
<custom_item>
 description: "Fortigate - Use non default admin access ports - 'HTTPS'"
 context: "config system global"
 not_expect: "set[\\s]+admin-sport[\\s]+443$"
</custom_item>
```

上記のチェックは、admin ポートが 443 に設定されている場合、不合格になります。

regex と expect

regex キーワードと **expect** キーワードの両方が指定されている場合、**regex** により、構成からすべての関連行が抽出され、**expect** が構成監査を実行します。**regex** と一致する行に **expect** に一致しない行がある場合、チェックは不合格になります。

```
<custom_item>
 description: "Fortigate - DNS - primary server"
 context: "config system dns"
 regex: "set[\\s]+primary"
 expect: "set[\\s]+primary[\\s]+1.1.1.1"
</custom_item>
```

regex と not_expect

regex キーワードと **not_expect** キーワードの両方が指定されている場合、**regex** により、構成からすべての関連行が抽出され、**not_expect** が構成監査を実行します。**regex** と一致する行に **not_expect** と一致する行がある場合、チェックは不合格になります。

```
<custom_item>
 description: "Fortigate - Disable insecure services - TELNET"
 context: "config system interface"
 regex: "set[\\s]+allowaccess"
 not_expect: "set[\\s]+allowaccess[\\s]+.*?(telnet[\\s]|telnet$)"
</custom_item>
```

上記のチェックは、構成で telnet が有効になっている場合、不合格になります。

context

context の概念は、すべてのコンプライアンス プラグインに適合するわけではありません。1 つの構成セクションに 1 つ以上の行を適用できるようにデバイス構成が構成されている場合、**.audit** のその特定セクションを監査するように **context** キーワードを使用します。たとえば、次の admin 設定例は、グローバル構成に構成/マップされます。

```
config system global
  set access-banner disable
  set admin-https-pki-required disable
  set admin-lockout-duration 60
  set admin-lockout-threshold 3
  set admin-maintainer enable
  set admin-port 80
.
```

cmd

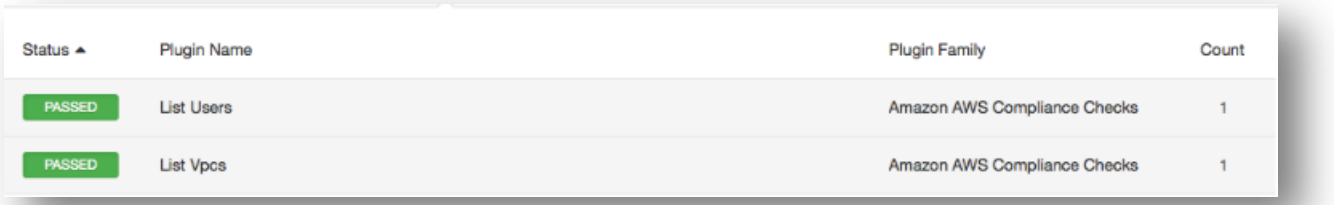
プラグインは **cmd** キーワードもサポートしています。このキーワードにより、**get** コマンドまたは **show** コマンドを実行して、レポートの結果出力を含めることができます。

```
<custom_item>
 description: "Fortigate - Review users with admin privileges"
 cmd: "get system admin"
 expect: ".+"
 severity: MEDIUM
</custom_item>
```

上記のチェックは、ターゲットから admin ユーザーを検出し、リストアップします。

Amazon AWS コンプライアンス ファイル リファレンス

Amazon AWS 監査には、実行中のインスタンス、ネットワーク ACL、ファイアウォール構成、アカウント構成、ユーザー リストなどに対するチェックが含まれます。リモート インスタンスの監査には、Amazon AWS アクセス キーの有効セットが必要です。Amazon AWS スキャンは、AWS が Web ベースのサービスであるため、ターゲットを指定しないという大きな点について、標準的な Nessus 監査と異なります。コンプライアンス結果は、他のコンプライアンス結果と似たものになります。



Status ▲	Plugin Name	Plugin Family	Count
PASSED	List Users	Amazon AWS Compliance Checks	1
PASSED	List Vpcs	Amazon AWS Compliance Checks	1

監査ファイルのシンタックス

Amazon AWS 構成チェック例を次に示します。

```
<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>
```

キーワード description、info、reference、および solution には任意のテキストを含めることができます。ユーザーは、**.audit** 内にチェック関連のメタデータを含めることができます。description 以外のキーワードはオプションです。

キーワード

次の表に、Amazon AWS コンプライアンス チェックにおける各キーワードの使用方法を示します。

キーワード	使用例とサポートされている設定
Type	このキーワードは、情報を戻すのにアクセスする API を指定します (ここでは IAM)。
description	"**description**" キーワードは、実行するチェックの簡単な説明を追加します。**description** フィールドには一意のテキストを指定して、**description** フィールドが重複するチェックは作成しないことを強く推奨します。Tenable の SecurityCenter は、一意のプラグイン ID 番号を **description** フィールドに基づいて自動的に生成します。
info	"**info**" キーワードは、実行するチェックに関する詳細な説明を追加するのに使用します。チェックの論理的根拠として、規則であったり、詳細情報が記載されている URL であったり、企業ポリシーなどを指定できます。複数の **info** フィールドを、テキストをパラグラフとしてフォーマットするために 1 行に 1 つずつ追加することができます。使用できる **info** フィールド数に制限はありません。

	 各 "**info**" タグは、改行なしで別の行に入力する必要があります。複数行が必要な場合は (フォーマット上の理由などにより)、"**info**" タグを追加します。 例: `info: "Review the the list of interfaces"` `info: "Disable unused interfaces"`
aws_action	このキーワードは、AWS セットアップに対して実行する Amazon API アクションを指定します。
xsl_stmt	このキーワードは、API リクエストの実行時に戻される XML ファイルに適用される XSLT を定義する方法を提供します。
regex	"**regex**" キーワードにより、構成項目設定の検索で、特定の正規表現への照合を行うことができます。 例: `regex: "set system syslog .+"` 次のメタ文字には特別な処理が必要です:+ \ * ( ) ^ これらの文字は、リテラルで解釈する場合は、2 つのバックスラッシュでエスケープするか、角括弧 "[]" で括ります。. ? " ' などの他の文字は、リテラルで解釈するには、バックスラッシュを 1 つだけ使用します。 これは、コンパイラがこれらの文字を処理する方法に関係します。 チェックに "**regex**" タグが設定され、"**expect**"、"**not_expect**"、"**number_of_lines**" タグは設定されていない場合、チェックは、regex と一致する全行をレポートします。
expect	このキーワードにより、"**regex**" タグに一致する構成項目を監査できます。また、"**regex**" タグが使用されていない場合、構成全体から "**expect**" 文字列が検索されます。 "**regex**" によって検出された構成行が "**expect**" タグと一致している場合、チェックは合格になります。"**regex**" が設定されていない場合は、"**expect**" 文字列が構成から検出された場合に合格になります。
not_expect	このキーワードにより、構成内に含まれてはならない構成項目を検索できます。 これは "**expect**" の反対の動作になります。"**regex**" によって検出された構成行が "**not_expect**" タグと一致しない場合、チェックは合格になります。"**regex**" タグが設定されていない場合は、構成に "**not_expect**" 文字列が検出されなければ合格です。

regex、**expect**、**not_expect** のどれも指定されていない場合、API クエリからの出力全体がレポートされます。

デバッグ

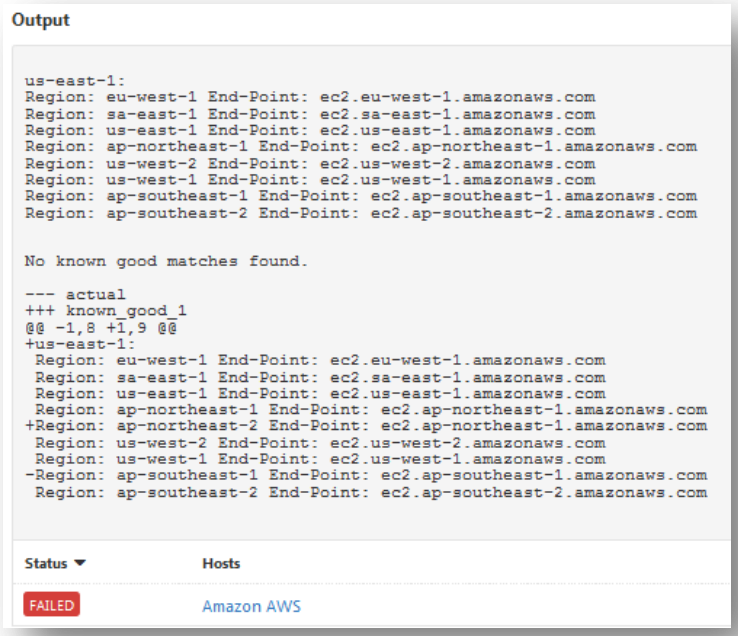
スキャンが適切に作動しないような問題が発生した場合、プラグインをデバッグ モードで実行させるデバッグ フラグを監査ファイルに新しく挿入できるようになりました。監査ファイル内のどこにでも `<debug/>` を追加できます。これにより、プラグインは、プラグイン エラーのトラブルシューティングに役立つ詳細情報を記録します。

既知の良好値監査

コンプライアンス監査は、一貫性と、既知の良好状態への適合を確認し、システムがこの状況に適合していることを繰り返し実証するためのものです。システムが既知の良好値から逸脱した場合に、これを把握することは重要です。こうすることによって、逸脱した結果生じた状況とその影響を分離させることが可能になります。これは通常、**regex**、**expect**、**not_expect**、およびその他同様のコンプライアンス ディレクティブの組み合わせにより実施します。この方法は多様であり、有効ではありますが、どうしてもテキストの 2 つの BLOB の比較には限界があります。どれだけうまく正規表現を設定しても、テキストの大きな BLOB と既知の良好値との比較を回避できる方法はありません。このことにより、"既知の良好値" に対してテキストの BLOB を比較するための機能を利用することができるようになっています。

この機能を利用するには、許容値を **known_good** キーワードにコピーする必要があります。複数の良好値を指定することもできますが、その場合は、コンマで区切ります。たとえば、次のとおりです。

```
<custom_item>
  Description    : "EC2: DescribeRegions - 'Regions that are currently available'"
  type           : EC2
  aws_action     : "DescribeRegions"
  xsl_stmt       : "<xsl:template match=\"/\">"
  xsl_stmt       : "<xsl:for-each select=\"//ec2:item\">"
  xsl_stmt       : "Region: <xsl:value-of select=\"ec2:regionName\"/> End-Point:
       <xsl:value-of select=\"ec2:regionEndpoint\"/><xsl:text>&#10;</xsl:text>"
  xsl_stmt       : "</xsl:for-each>"
  xsl_stmt       : "</xsl:template>"
  known good     : 'us-east-1:
Region: eu-west-1 End-Point: ec2.eu-west-1.amazonaws.com
Region: sa-east-1 End-Point: ec2.sa-east-1.amazonaws.com
Region: us-east-1 End-Point: ec2.us-east-1.amazonaws.com
Region: ap-northeast-1 End-Point: ec2.ap-northeast-1.amazonaws.com
Region: ap-northeast-2 End-Point: ec2.ap-northeast-1.amazonaws.com
Region: us-west-2 End-Point: ec2.us-west-2.amazonaws.com
Region: us-west-1 End-Point: ec2.us-west-1.amazonaws.com
Region: ap-southeast-2 End-Point: ec2.ap-southeast-2.amazonaws.com'
</custom_item>
```



出力には、監査しやすさのための差分が示されています。

ユース ケース

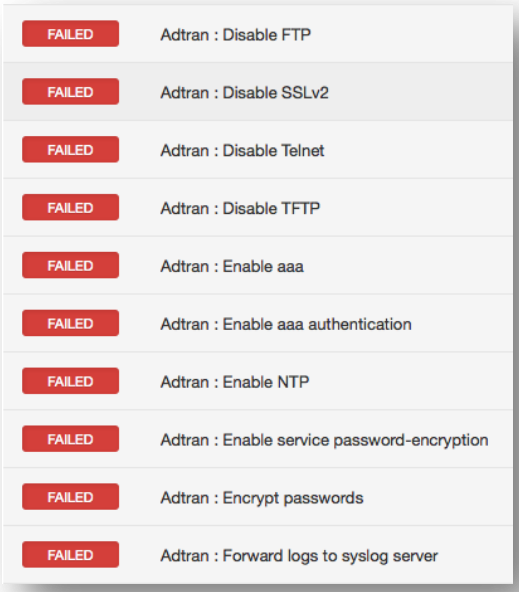
この機能の最も便利なユース ケースの 1 つとして、既知の全良好値を使用した "ゴールド スタンダード" 監査を作成するという方法があります。たとえば、ターゲットに対して要件を満たすためのスキャンを実行し、**.nessus** ファイルから "known_good" 値を取得して、監査ファイルを更新し、"すべて合格" の結果を出すようにもう一度スキャンを実行することができます。

その他

- **known_good** を使用すると **expect** と **not_expect** は上書きされますが、**regex** は残ります。したがって、**regex** が指定されている場合、regex で絞り込まれたデータに対する比較が行われます。
- 1 つのルールで複数の **known_good** を指定できますが、それぞれコンマで区切る必要があります。
- この機能は **.inc** ファイルのスタンドアロン機能として実装されているので、どの Nessus プラグイン内でも簡単に使用できます。

Adtran AOS コンプライアンス ファイル リファレンス

Adtran AOS 監査には、パスワード ポリシー、有効サービス、セキュリティ保護されていないサービス構成、認証、ログ設定と監査設定、および SNMP & NTP 構成設定に対するチェックが含まれます。完全権限を持つ root または管理者の有効な SSH 資格情報が必要です。



シンタックス

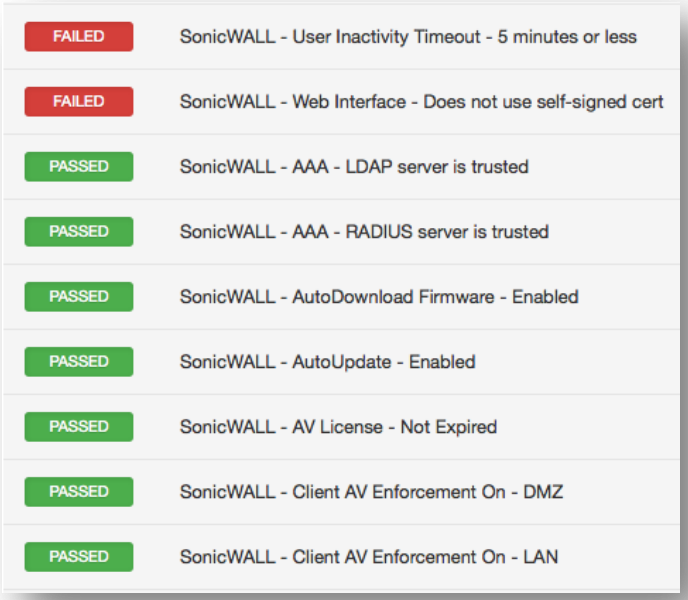
このプラグインと監査用のシンタックスは次のとおりです。

```
<custom_item>
 description: "Adtran : Disable FTP"
info: "Disable ftp server, if not required."
 not_expect: "^ip ftp server"
 solution: "Do disable FTP Server, run the following command :\n
     no ip ftp server"
 reference: "PCI|2.2.3,SANS-CSC|10,CSF|PR.DS-2,800-53|AC-17,800-53|SC-9"
</custom_item>
```

SonicWALL SonicOS コンプライアンス ファイル リファレンス

SonicWALL SonicOS 監査には、SSL 構成、パスワード ポリシー、バナー構成、管理アクセス ポート、無通信タイムアウト設定、フラッド防止設定、クライアント AV 強制ポリシー、ログ設定と監査設定、有効セキュリティ サービス、ゲートウェイ ウイルス対策構成、承認設定と認証設定、および侵入保護サービス構成に対するチェックが含まれます。

SonicWall への SSH 導入は、拡張テストに基づくと信頼できない場合があります。SSH API が監査に合格しなかった場合、オフライン構成監査を推奨します。



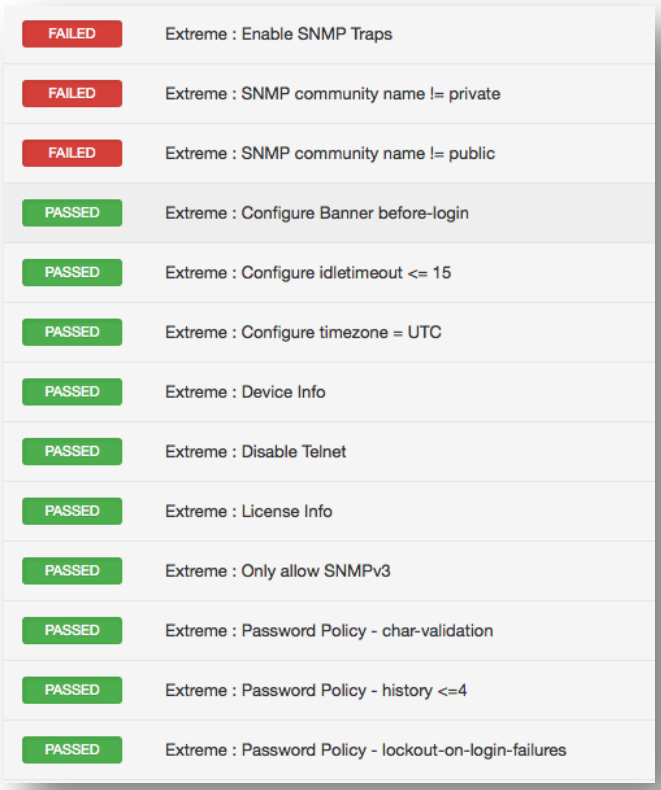
シンタックス

このプラグインと監査用のシンタックスは次のとおりです。

```
<custom_item>
 description: "SonicWALL - Disable insecure services - HTTP"
 info: "HTTP is insecure by nature as it sends all traffic across the wire in clear
       text."
 solution: "Navigate to network->interfaces. Configure each interface by unchecking
       the http management box."
 reference: "800-53|CM-7,SANS-CSC|11,SANS-CSC|10,PCI|2.2.3,CSF|PR.PT-3,800-53|CM-6"
 cmd: "show interface all"
regex: "http[\\s]mgmt"
 not_expect: "http[\\s]mgmt[\\s]+on"
</custom_item>
```

Extreme ExtremeXOS コンプライアンス ファイル リファレンス

Extreme ExtremeXOS 監査には、パスワード ポリシー、バナー構成、無通信タイムアウト設定、ログ設定と監査設定、セキュリティ保護されていないサービス、デバイス ライセンス情報、および SNMP 設定に対するチェックが含まれます。



シンタックス

このプラグインと監査用のシンタックスは次のとおりです。

```
<custom_item>
 description: "Extreme : Password Policy - min-length >= 8"
 info: "Do not allow password lengths less than 8 characters"
 expect: "configure account all password-policy min-length ([8-9]|[1-9][0-9]+)"
 solution: "Run the following command to enforce min password length :\n
 configure account all password-policy min-length 8"
 reference: "SANS-
       CSC|10,HIPAA|164.308(a)(5)(ii)(D),PCI|2.2.4,PCI|8.2.3,COBIT5|BAI10.01,800-
       53|CM-2"
</custom_item>
```

データベース構成監査コンプライアンス ファイル リファレンス

ここでは、データベース コンプライアンス チェックのフォーマットと関数、および各設定の論理的根拠について説明します。

引用符の用法:

監査フィールドを囲む一重引用符と二重引用符は、次の 2 つの場合を除いてどちらでも使用可能です。

1. CRLF など、文字をリテラルに解釈する必要がある特殊フィールドを含む Windows コンプライアンス チェックでは、

一重引用符を使用します。文字列として解釈する組み込みフィールドはエスケープする必要があります。

たとえば、次のとおりです。
```
expect: 'First line\r\nSecond line\r\nJohn\'s Line'
```

2. 二重引用符は WindowsFiles の "include_paths" と "exclude_paths" を使用する際に必要です。

どのフィールドタイプ (description、value_data、regex など) においても、一重引用符または二重引用符を含む文字列を使用する場合は、次のいずれかの方法を用います。

a. 外側の引用符とは異なる引用符タイプを使用する。

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

b. 組み込み引用符はバックスラッシュでエスケープする (二重引用符のみ)。

例:
```
expect: "\"Text to be searched\""
```

チェック タイプ

すべてのデータベース コンプライアンス チェックは、**check_type** カプセルで囲われ、"Database" 指定を含んでいる必要があります。これは、この **.audit** ファイルが、データベース用のファイルであることを識別するためのものです。**check_type** フィールドには、次の 2 つのパラメータが必要です。

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

監査対象にできるデータベース タイプは次のとおりです。

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

version は現在、常に "1" に設定されます。

例:

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

キーワード

次の表に、データベース コンプライアンス チェックにおける各キーワードの使用方法を示します。

キーワード	使用例とサポートされている設定
type	SQL_POLICY

description	このキーワードは、実行するチェックの簡単な説明を追加します。**description** フィールドには一意のテキストを指定して、description フィールドが重複するチェックは作成しないことを強く推奨します。Tenable の SecurityCenter は、一意のプラグイン ID 番号を **description** フィールドに基づいて自動的に生成します。 例: `description: "DBMS Password Complexity"`
info	このキーワードは、実行するチェックに関する詳細な説明を追加するのに使用します。このキーワードにより、規則、URL、企業ポリシー、または設定が必要なその他の理由を示します。複数の **info** フィールドを、テキストをパラグラフとしてフォーマットするために 1 行に 1 つずつ追加することができます。使用できる **info** フィールド数に制限はありません。 例: `info: "Checking that \"password complexity\" requirements are enforced for systemsusing SQL Server authentication."`
sql_request	このキーワードは、データベースに送信する実際の **SQL** リクエストを指定するのに使用します。コンマ区切りリクエスト/戻り値を使用して、データ配列を要求し、**SQL** リクエストから返すことができます。 例: `sql_request: "select name from sys.sql_logins where type = 'S' and is_policy_checked <> '1'"` 例: `sql_request: "select name, value_in_use from sys.configurations where name = 'clr enabled'"`
sql_types	このキーワードには、2 つの使用可能なオプション **POLICY_VARCHAR** と **POLICY_INTEGER** が含まれています。0 ～ 2147483647 の数値に対しては **POLICY_INTEGER** を、その他の戻り値タイプに対しては **POLICY_VARCHAR** を使用します。 例: `sql_types: POLICY_VARCHAR` 例: `sql_types: POLICY_VARCHAR,POLICY_INTEGER` 複数の戻り項目に対しては、コンマ区切りリストで **sql_types** を構成し、各 **SQL** 戻り結果のデータ型を指定します。上記の例では、**SQL** クエリからの最初の戻り値は **varchar** で、2 つ目の戻り値は整数になっています。
sql_expect	このキーワードは、**SQL** リクエストからの予定戻り値を決定します。**NULL** または "0" を含む正確な値を指定できます。さらに、**POLICY_VARCHAR sql_types** には正規表現が必要です。 例: `sql_expect: regex:"^.+Failure" || regex:"^.+ALL"` 例: `sql_expect: NULL`

	例: `sql_expect: 0 || "0"` 二重引用符は、整数戻り値にはオプションです。 例: `sql_expect: "clr enabled",0` データ配列が **SQL** リクエストから返され、**sql_expect** フィールドにコンマ区切り形式で挿入される場合もあります。

コマンド ライン例

ここでは、データベース コンプライアンス チェック用に使用される一般的な監査例をいくつか紹介します。**nasl** コマンド ライン バイナリは、実行中に簡単に監査をテストするための方法として使用されます。以下の各 **.audit** ファイルは、Nessus 4 または SecurityCenter スキャンポリシーに簡単にドロップできます。1 つのシステムのクイック監査については、コマンド ライン テストが効率的です。**/opt/nessus/bin** ディレクトリから毎回実行するコマンドは次のとおりです。

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

<IP> は、監査対象システムの IP アドレスです。

監査対象のデータベースのタイプに応じて、使用する監査ファイル以外のパラメータの入力が必要になる場合があります。たとえば、Oracle 監査では、データベース SID と Oracle ログイン タイプが必要になります。

```
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
```

正しいデータベース ログイン パラメータについては、データベース管理者に問い合わせてください。

例 1:有効期限のないログインを検索する

有効期限のない SQL Server ログインを検索する簡単な **.audit** ファイルの例を次に示します。検出されると、監査は、違反ログインとともにエラー メッセージを表示します。

```
<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>
```

このコマンドを実行すると、準拠システムであれば、次の出力が生成されます。

```
"Login expiration check": [PASSED]
```

データベース ログインに有効期限があることは、通常、コンプライアンス要件になっています。

不合格監査では、次の出力が返されます。

```
"Login expiration check": [FAILED]

Database logins with no expiration date pose a security threat.

Remote value:

"distributor_admin"

Policy value:

NULL
```

この出力では、"distributor_admin" アカウントには有効期限が構成されておらず、システム セキュリティ ポリシーに対するチェックが必要であることが示されています。

例 2:不正ストアド プロシージャの有効状態をチェックする

この監査は、ストアド プロシージャ "SQL Mail XPs" が有効かどうかをチェックします。外部ストアド プロシージャは、システムによってはセキュリティの脅威になり得るので、無効にしておくことが必要な場合があります。

```
<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>
```

上記のチェックでは、"SQL Mail XPs" ストアド プロシージャが無効 (value_in_use = 0) の場合、合格になります。そうでない場合、不合格になります。

例 3:混合結果 sql_type のデータベースの状態をチェックする

場合によっては、コンプライアンス データベース クエリは、複数のデータ型の結果を返す複数のデータ リクエストを必要とします。次の監査例は、データ型の混合をチェックし、どのように出力解析が可能かを実証します。

```
<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>
```

sql_request 値、**sql_types** 値、および **sql_expect** 値にすべて、コンマ区切り値が含まれていることに注意してください。

条件

データベース ポリシー内に **if/then/else** 論理を定義することができます。これにより、監査が合格した場合、合格/不合格の結果ではなく、警告メッセージを返すことができます。

条件を実行するシンタックスは次のとおりです。

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

例:

```
<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>
```

"サイレント" チェックなので、条件の失敗または成功はレポートに表示されません。

条件タイプは "**and**" または "**or**" のいずれかになります。

Unix 構成監査コンプライアンス ファイル リファレンス

ここでは、Unix コンプライアンス チェックのビルトイン関数、および各設定の論理的根拠について説明します。

引用符の用法:

監査フィールドを囲む一重引用符と二重引用符は、次の 2 つの場合を除いてどちらでも使用可能です。

1. CRLF など、文字をリテラルに解釈する必要がある特殊フィールドを含む Windows コンプライアンス チェックでは、一重引用符を使用します。文字列として解釈する組み込みフィールドはエスケープする必要があります。

たとえば、次のとおりです。

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

2. 二重引用符は WindowsFiles の "include_paths" と "exclude_paths" を使用する際に必要です。

どのフィールドタイプ (description、value_data、regex など) においても、一重引用符または二重引用符を含む文字列を使用する場合は、次のいずれかの方法を用います。

a. 外側の引用符とは異なる引用符タイプを使用する。

例:

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

b. 組み込み引用符はバックスラッシュでエスケープする (二重引用符のみ)。

例:

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

チェック タイプ

すべての Unix コンプライアンス チェックは、"**check_type**" カプセルで囲われ、"Unix" 指定を含んでいる必要があります。付録 A には、"Unix" の **check_type** 設定で始まり、"**</check_type>**" タグで終了する Unix コンプライアンス チェック例が紹介されています。

これは、この **.audit** ファイルが、Windows (またはその他のプラットフォーム) 用のファイルであることを識別するためのものです。

ファイルは SSH から、Nessus サーバー上のメモリ バッファに読み込まれ、バッファがコンプライアンス チェック用に処理されます。

キーワード

次の表に、Unix コンプライアンス チェックにおける各キーワードの使用方法を示します。

キーワード	使用例とサポートされている設定
attr	このキーワードは、FILE_CHECK および FILE_CHECK_NOT で使用して、ファイルに関連付けられているファイル属性を監査します。ファイルの属性の構成について詳しくは、chattr(1) man ページを参照してください。
comment	このフィールドは、description フィールドには適さない補足情報を追加するのに使用されます。 例: `comment: (CWD - Current working directory)`
description	このキーワードは、実行するチェックを簡単に説明するのに使用します。**description** フィールドは一意である必要があり、2 つのチェックが同じ description フィールドを持つことはできません。Tenable の SecurityCenter は、一意のプラグイン ID 番号を **description** フィールドに基づいて自動的に生成します。 例: `description: "Permission and ownership check for /etc/at.allow"`
dont_echo_cmd	このキーワードは、"CMD_EXEC" Unix コンプライアンス チェック監査で使用され、チェックによって実行された実際のコマンドを出力から省略するよう指示します。コマンドの結果のみが表示されます。 例: `dont_echo_cmd: YES`
except	このキーワードは、チェックから特定のユーザー、サービス、およびファイルを除外するのに使用されます。 例: `except: "guest"` 複数のユーザー アカウントはパイプで区切ります。 例: `except: "guest" | "guest1" | "guest2"`
expect	このキーワードは、**regex** とともに使用します。これにより、ファイル内の特定値を検索する機能が提供されます。

	例: `<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	このキーワードは、パーミッションとオーナー権限設定をチェックするファイルの絶対パスまたは相対パスを示すのに使用されます。 例: `file: "/etc/inet/inetd.conf"` `file: "~/inetd.conf"` **file** 値としては glob を指定することもできます。 例: `file: "/var/log/*"` この機能は、FILE_CHECK、FILE_CONTENT_CHECK、FILE_CHECK_NOT、または FILE_CONTENT_CHECK_NOT を使用して特定ディレクトリ内のすべてのファイルに対してパーミッションまたはコンテンツを監査する必要がある場合に特に便利です。
file_type	このキーワードは検索対象のファイルのタイプを示します。 サポートされるファイル タイプは次のとおりです。 • b - ブロック型 (バッファ) 特殊 • c - キャラクタ型 (バッファなし) 特殊 • d - ディレクトリ • p - 名前付きパイプ (FIFO) • f - レギュラー ファイル 例: `file_type: "f"` 複数のファイル タイプは同じ文字列内にパイプ記号で連結できます。 例: `file_type: "c|b"`
group	このキーワードは、**file** キーワードとともに使用して、ファイルのグループを指定します。**group** キーワードには、オーナーのないファイルを検索できる "none" 値を指定できます。 例: `group: "root"` グループは、次のシンタックスを使用して論理 "OR" 条件で指定することもできます。

	group: "root" || "bin" || "sys"
ignore	このキーワードは、検索から指定ファイルを無視するのに使用します。このキーワードは、FILE_CHECK 、 FILE_CHECK_NOT 、 FILE_CONTENT_CHECK 、 および FILE_CONTENT_CHECK_NOT の各チェック タイプに対して使用できます。 例: # 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	このキーワードは、実行するチェックに関する詳細な説明を追加するのに使用します。このキーワードにより、規則、URL、企業ポリシー、または設定が必要な理由を示します。複数の **info** フィールドを、テキストをパラグラフとしてフォーマットするために 1 行に 1 つずつ追加することができます。使用できる **info** フィールド数に制限はありません。 例: info: "ref. CIS_AIX_Benchmark_v1.0.1.pdf ch 1, pg 28-29."
levels	このキーワードは、CHKCONFIG で使用し、サービスを実行する実行レベルを指定します。実行レベルはすべて単一文字列で指定する必要があります。たとえば、サービス "sendmail" を実行レベル 1、2、および 3 で実行する必要がある場合、CHKCONFIG チェック内の対応する **levels** 値は次のようになります。 levels: "123"
mask	このキーワードは、特定のユーザー、グループ、またはその他メンバーに対して使用してはならないパーミッションを指定できるモードの反対になります。*正確な*パーミッション値をチェックする **mode** とは異なり、**mask** 監査はより範囲が広く、ファイルまたはディレクトリが、**mask** によって指定された値と同じ、またはよりセキュリティが高いレベルにあるかどうかをチェックします。たとえば、**mode** が、監査期待値 644 と一致しないということで、640 のパーミッションで不合格となった場合でも、**mask** では、640 は "よりセキュリティが高い" という判断になり、監査には合格します。 例: mask: 022 これは、オーナーにはすべてのパーミッションが了承され、グループとその他のメンバーには書き込みパーミッションを提供しないことを示しています。**mask** 値 "7" は、その特定のオーナー、グループ、またはその他のメンバーにパーミッションがないことを意味します。
md5	このキーワードは FILE_CHECK と FILE_CHECK_NOT で使用され、ファイルの MD5 がポリシーの設定内容に確実に適用されるようにします。 例: <custom_item> type: FILE_CHECK description: "/etc/passwd has the proper md5 set"

	required: YES file: "/etc/passwd" md5: "ce35dc081fd848763cab2cfd442f8c22" </custom_item>
mode	このキーワードは、ファイル/フォルダのパーミッション セットを示します。**mode** キーワードは、文字列または 8 進数フォーマットで表現できます。 例: mode: "-rw-r--r--" mode: "644" mode: "7644"
name	このキーワードは PROCESS_CHECK でプロセス名の識別に使用されます。 例: name: "syslogd"
operator	このキーワードは、RPM_CHECK と PKG_CHECK に関連して使用し、インストールした RPM パッケージのバージョンに基づいてチェックを合格/不合格にする条件を指定します。次の値を指定できます。 • **lt** (未満) • **lte** (以下) • **gte** (以上) • **gt** (より大きい) • **eq** (等しい) 例: operator: "lt"
owner	このキーワードは、**file** キーワードとともに使用して、ファイルのオーナーを指定します。**owner** キーワードには、オーナーのないファイルを検索できる "none" 値を指定できます。 例: owner: "root" オーナーは、次のシンタックスを使用して論理 "OR" 条件で指定することもできます。 owner: "root" || "bin" || "adm"
reference	このキーワードにより、**.audit** に相互参照を含めることができます。フォーマットは "ref|ref-id1,ref|ref-id2" です。 例: 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	このキーワードは、特定の正規表現と一致するファイルを検索できます。 例: regex: ".*LogLevel=9$" 次のメタ文字には特別な処理が必要です:+ \ * ( ) ^

	これらの文字は、リテラルで解釈する場合は、2 つのバックスラッシュでエスケープするか、角括弧 "[]" で括ります。. ? " ' などの他の文字は、リテラルで解釈するには、バックスラッシュを 1 つだけ使用します。 これは、コンパイラがこれらの文字を処理する方法に関係します。
required	このキーワードは、監査対象項目がリモート システム上に存在する必要があるかないかを指定するのに使用します。たとえば、**required** が "NO" で、チェック **タイプ** が "FILE_CHECK" の場合、ファイルが存在して、パーミッションが **.audit** ファイルで指定されているとおりか、ファイルが存在していない場合に、チェックは合格になります。**required** が "YES" の場合は、上記のチェックは不合格になる場合があります。
rpm	このキーワードは、RPM_CHECK で使用し、検索する RPM を指定します。 例: `<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	このキーワードは、ファイル システム内の検索可能場所を指定するのに使用できます。 例: `search_locations: "/bin"` 複数の検索場所はパイプで区切ります。 例: `search_locations: "/bin" | "/etc/init.d" | "/etc/rc0.d"`
see_also	このキーワードにより、参照リンクを含めることができます。 例: `see_also: "https://benchmarks.cisecurity.org/tools2/linux/CIS_Redhat_Linux_5_Benchmark_v2.0.0.pdf"`
service	このキーワードは CHKCONFIG、XINETD_SVC、および SVC_PROP で使用し、監査対象のサービスを指定します。 例: `<custom_item>` `type: CHKCONFIG` `description: "2.1 Disable Standard Services - Check if cups is disabled"` `service: "cups"` `levels: "123456"` `status: OFF` `</custom_item>`
severity	**<item>**、**<custom_item>** かかわらずどのテストにでも、"**severity**" フラグを追加して、"LOW"、"MEDIUM"、または "HIGH" に設定できます。デフォルトでは、不適合結果は

	"HIGH" として表示されます。 例: `severity: MEDIUM`
solution	このキーワードにより、必要に応じて "ソリューション" テキストを含めることができます。 例: `solution: "Remove this file, if its not required"`
status	このキーワードは PROCESS_CHECK、CHKCONFIG、および XINETD_SVC で使用し、特定のホスト上のサービスが実行する必要があるのか無効にする必要があるのかを判断します。**status** キーワードは 2 つの値 "ON" または "OFF" のいずれかになります。 例: `status: ON` `status: OFF`
system	このキーワードは、チェックを実行するシステムのタイプを指定します。  "**system**" キーワードは "**custom_item**" チェックにのみ適用可能です。ビルトイン "**item**" チェックには使用できません。 使用可能な値は、ターゲット OS で "**uname**" コマンドによって返される値です。たとえば、Solaris 上では値は "SunOS"、Mac OS X では "Darwin"、FreeBSD では "FreeBSD" になります。 例: `system: "SunOS"`
timeout	このキーワードは CMD_EXEC で使用して、指定コマンドがタイムアウトするまで実行可能な秒数を指定します。このキーワードは、Unix "**find**" コマンドなど、完了するまでの時間を拡張する必要がある場合に便利です。このキーワードを指定しない場合、CMD_EXEC 監査のデフォルト タイムアウトは 5 分です。 例: `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	**value** キーワードは、システムの設定がポリシー値に準拠しているかどうかを確認するのに便利です。

	例: `value: "90..max"` **value** キーワードは、範囲 [数値..max] で指定できます。値が指定値と "max" 値の間にあれば、合格になります。

カスタム項目

カスタム項目は、前記のキーワードに基づいて定義される 1 つの完全なチェックです。カスタム項目の一覧を以下に示します。各チェックが "**<custom_item>**" タグで始まり、"**</custom_item>**" で終わります。これらのタグ内に、コンプライアンス チェック パーサーによって解釈され、チェックを実行する 1 つ以上のキーワードが配置されます。

カスタム監査チェックでは、終了タグとして "**</custom_item>**" と "**</item>**" のどちらでも使用できます。

AUDIT_XML

"AUDIT_XML" 監査チェックでは、まず XSLT を適用し、関連データを抽出し、**regex**、**expect**、**not_expect** の各キーワードに基づいてコンプライアンスを行うことにより XML ファイルを検証および監査できます (詳細については、付録 C を参照)。このチェックは、4 つ以上のキーワードで構成されます。type、description、file、xsl_stmt の各ディレクティブ (必須) の後に、**regex**、**expect**、**not_expect** の各キーワードが続いて、これらによりコンテンツを監査します。

例:

```
<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>
```

file キーワードにはワイルドカードを使用できます。たとえば、次のとおりです。

```
<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

"CHKCONFIG" 監査チェックは、監査対象のリモート Red Hat システム上の "**chkconfig**" ユーティリティと対話します。このチェックは、5 つの必須キーワード **type**、**description**、**service**、**levels**、および **status** で構成されます。

CHKCONFIG 監査は、Red Hat システム、または Fedora などの Red Hat 派生システムでのみ実行可能です。

例:

```
<custom_item>
type:CHKCONFIG
 description:"Make sure that xinetd is disabled"
 service:"xinetd"
 levels:"123456"
 status:OFF
</custom_item>
```

CMD_EXEC

リモート ホスト上でコマンドを実行して、出力が期待値と一致しているかチェックできます。このチェック タイプは、Unix のさまざまな構成間で必ず移植できるわけではないので、注意を要します。

quiet キーワードは、不合格になったコマンド出力を表示**しない**ためのものです。これは "YES" または "NO" に設定できます。デフォルトでは、"NO" に設定され、コマンド結果が表示されます。同様に、"**dont_echo_cmd**" キーワードにより、コマンドそのものではなくコマンド結果を出力することによって、結果を絞り込むことができます。

nosudo キーワードは、"YES" に設定することによって、sudo を使用してコマンドを実行**しない**ように指定できます。デフォルトでは、"NO" に設定され、sudo は、構成に応じて常に使用できる状態になっています。

例:

```
<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

Unix コンプライアンス監査は、通常、特定のファイルの存在と設定をテストします。"FILE_CHECK" 監査は、4 つ以上のキーワードを使用して、これらチェックの仕様を定義できます。キーワード **type**、**description**、および **file** は必須です。この後に 1 つ以上のチェックが続きます。現在のシンタックスは、オーナー パーミッション、グループ パーミッション、およびファイル パーミッションのチェックをサポートしています。

FILE_CHECK では glob を使用できます (**/var/log/*** など)。ただし、glob は、ディレクトリではなくファイルにのみ展開されます。glob を指定し、検索から 1 つ以上の一致ファイルを無視する必要があった場合、"**ignore**" キーワードを使用して、無視するファイルを指定します。

許可されるキーワードは次のとおりです。

```
uid: Numeric User ID (e.g., 0)
gid: Numeric Group ID (e.g., 500)
check_uneveness: YES
system: System type (e.g., Linux)
description: Text description of the file check
file: Full path and file to check (e.g., /etc/sysconfig/sendmail)
owner: Owner of the file (e.g., root)
group: Group owner of the file (e.g., bin)
mode: Permission mode (e.g., 644)
mask: File umask (e.g., 133)
md5: The MD5 hash of a file (e.g., 88d3dbe3760775a00b900a850b170fcd)
ignore: A file to ignore (e.g., /var/log/secure)
attr: A file attribute (e.g., ----i--------)
```

"group" または "other" に "owner" 以上のパーミッションがある場合、または "other" に "group" 以上のパーミッションがある場合、ファイル パーミッションは不均一だと考えられます。

次に例を示します。

```
<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

"FILE_CHECK_NOT" 監査は 3 つ以上のキーワードで構成されます。キーワード **type**、**description**、および **file** は必須です。この後に 1 つ以上のチェックが続きます。現在のシンタックスは、オーナー パーミッション、グループ パーミッション、およびファイル パーミッションのチェックをサポートしています。FILE_CHECK 監査と同様、ファイル glob を指定する場合、"**ignore**" キーワードを使用して、1 つ以上のファイルを無視できます。

この機能は FILE_CHECK の反対です。ファイルが存在しない場合、またはモードがチェックで定義されているモードと同じである場合、ポリシーは不合格になります。

FILE_CHECK_NOT では glob を使用できます (**/var/log/*** など)。ただし、glob は、ディレクトリではなくファイルにのみ展開されます。

次に例を示します。

```
<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

ファイルの存在および設定のテストと同様、テキスト ファイルのコンテンツも分析できます。正規表現を使用して、既存コンテンツについて 1 つ以上の場所を検索できます。"**ignore**" キーワードを使用して、指定の検索場所から 1 つ以上のファイルを無視できます。

string_required フィールドにより、検索対象の監査文字列の存在を必須とするか、必須としないかを指定できます。このオプションを設定しない場合、必須が前提となります。**file_required** フィールドにより、監査対象ファイルの存在を必須とするかしないかを指定できます。このオプションを設定しない場合、必須が前提となります。

次に例を示します。

```
<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>
```

ファイル パラメータに "~" を追加することにより、FILE_CONTENT_CHECK が不適合コンテンツのスキャンをユーザーの home ディレクトリに対して実施するよう指定できます。

```
<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

この監査は、**regex** フィールドの正規表現と一致するかどうかファイル コンテンツを検証します。この機能は FILE_CONTENT_CHECK の反対です。つまり、正規表現の一致がファイル内に**ある**場合、ポリシーは不合格になります。"**ignore**" キーワードを使用して、指定の検索場所から 1 つ以上のファイルを無視できます。

このポリシー項目は、ファイルに正規表現 regex が含まれ、この正規表現が **expect** と一致していないことをチェックします。

許可されるタイプは次のとおりです。

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

このチェックでは、**regex** と **expect** の両方が指定されている必要があります。

次に例を示します。

```
<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

"GRAMMAR_CHECK" 監査チェックは、ファイル コンテンツを検証し、大まかに定義された文法との照合 (1 つ以上の正規表現文で構成) を行います。ターゲット ファイル内の **1** 行が正規表現文のいずれかと一致しない場合、不合格になります。

例:

```
<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

"MACOSX_DEFAULTS_READ" 監査チェックは、MAC OS X のデフォルト システム値を検証します。このチェックは、設定されているプロパティに応じて、動作が異なります。

plist_user が "all" に設定されている場合、すべてのユーザー設定が監査対象となり、それ以外の場合は、指定のユーザー設定のみが監査対象になります。

plist_user プロパティの設定に加え、**byhost** プロパティが YES に設定されている場合、次のクエリが実行されます。

```
/usr/bin/defaults -currentHost read /Users/foo/Library/Preferences/ByHost/plist_name
       plist_item
```

plist_user プロパティが設定され、**byhost** プロパティが設定されていない場合、次のクエリが実行されます。

```
/usr/bin/defaults -currentHost read /Users/foo/Library/Preferences/plist_name
       plist_item
```

plist_user プロパティも **byhost** プロパティも設定されていない場合、次のクエリが実行されます。

```
/usr/bin/defaults -currentHost read plist_name plist_item
```

サポートされているプロパティは次のとおりです。

```
plist_name : the plist we want to query. for e.g. com.apple.digihub
plist_item : The plist item to be audited. for e.g. com.apple.digihub.blank.cd.appeared
plist_option : CANNOT_BE_NULL. If this is set to CANNOT_BE_NULL, the check fails if the
setting being audited is not set.
byhost : YES. Setting byhost to YES, results in a slightly different query.
```

例:

```
<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

"PKG_CHECK" 監査チェックは、SunOS システムに対して **pkgchk** を実行します。**pkg** キーワードを使用して、検索対象のパッケージを指定し、**operator** キーワードにより、インストール済みパッケージのバージョンに応じてチェックの合格/不合格を決定する条件を指定します。

例:

```
<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

ファイル チェックと同様、監査対象 Unix プラットフォームもプロセス実行用にテストできます。"**ps**" コマンドが実行され、実行中のプロセス リストが取得されます。

例:

```
<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

"RPM_CHECK" 監査チェックを使用して、リモート システムにインストールされている RPM パッケージのバージョン番号をチェックできます。このチェックは、4 つの必須キーワード **type**、**description**、**rpm**、**operator** と、オプション キーワードである **required** で構成されます。**rpm** キーワードを使用して、検索対象のパッケージを指定し、**operator** キーワードにより、インストール済みパッケージのバージョンに応じてチェックの合格/不合格を決定する条件を指定します。

RPM チェックの使用は、Linux ディストリビューション間で移植できません。したがって、RPM_CHECK の使用は移植できません。

iproute-2.4.7-10 がインストールされていることを前提とした例を次に示します。

```
<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

"SVC_PROP" 監査チェックは、Solaris 10 システム上の "**svcprop -p**" ツールと対話できます。このチェックは、特定のサービスと関連付けられているプロパティの問い合わせを行う場合に使用できます。**service** キーワードは、監査対象のサービスを指定するのに使用します。**property** キーワードは、問い合わせるプロパティの名前を指定します。**value** キーワードは、プロパティの期待値になります。この期待値は正規表現でも構いません。

svcprop_option フィールドにより、検索対象の監査文字列の存在を必須とするか、必須としないかを指定できます。このフィールドは引数として CAN_BE_NULL または CANNOT_BE_NULL にアクセスします。

例:

```
<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

"XINETD_SVC" 監査チェックは、xinetd サービスのスタートアップ ステータスを監査するのに使用されます。このチェックは、4 つの必須キーワード **type**、**description**、**service**、および **status** で構成されます。

これは、Red Hat システム、または Fedora などの Red Hat 派生システムでのみ実行可能です。

例:

```
<custom_item>
type:XINETD_SVC
 description:"Make sure that telnet is disabled"
 service:"telnet"
 status:OFF
</custom_item>
```

ビルトイン チェック

上記のチェックに含まれていないチェックは、NASL でカスタム名として記述する必要があります。このようなチェックはすべて "ビルトイン" カテゴリに分類されます。各チェックが "**<item>**" タグで始まり、"**</item>**" で終わります。これらのタグ内に、コンプライアンス チェック パーサーによって解釈され、チェックを実行する 1 つ以上のキーワードが配置されます。使用可能なチェックの一覧を以下に示します。

"**system**" キーワードは、ビルトイン チェックでは使用できません。使用すると、シンタックス エラーになります。

パスワード管理

以下の例では、監査値データで使用する文字列ではなく、整数値を表すのに <min> と <max> を使用しています。

正確な最小値または最大値がわからない場合は、整数値に対して文字列 "Min" または "Max" を代入します。

min_password_length

使用法

```
<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>
```

このビルトイン チェックは、リモート システム上の最小パスワード長が "**<min>..<max>**" の範囲にあることを確認します。最小パスワード長を設定しておくと、ユーザーによる複雑なパスワードの作成を促進できます。

オペレーティング システム	実装
Linux	最小パスワード長は **/etc/login.defs** に PASS_MIN_LEN として定義されています。

Solaris	最小パスワード長は、**/etc/default/passwd** 内に PASSLENGTH として定義されています。これは、パスワード最大長も制御します。
HP-UX	最小パスワード長は、**/etc/default/security** 内に MIN_PASSWORD_LENGTH として定義されています。
Mac OS X	最小パスワード長は、コマンド **pwpolicy** を使用して定義されたローカル ポリシー内の "minChar" として定義されています。

例:

```
 <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

使用法

```
<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>
```

このビルトイン関数は、パスワードの最大有効期間 (ユーザーがパスワードの変更を強制されるまでの時間) が定義された範囲内であることを確認します。

パスワードの最大有効期間を設定することで、同じパスワードが何年も使用されないようにすることができます。パスワードの変更を頻繁に行うことによって、パスワードを取得できた攻撃者がそのパスワードを無制限に使えないようにします。

オペレーティング システム	実装
Linux	変数 PASS_MAX_DAYS が **/etc/login.defs** に定義されています。
Solaris	**/etc/default/passwd** 内の変数 MAXWEEKS は、パスワードの最大有効期間 (週単位) を定義します。
HP-UX	この値は、**/etc/default/security** 内の変数 PASSWORD_MAXDAYS によって制御されます。
Mac OS X	パスワード ポリシー (**pwpolicy** ツールから設定) のオプション "maxMinutesUntilChangePassword" を使用して、この値が設定されます。

例:

```
 <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

使用法

```
<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>
```

このビルトイン関数は、パスワードの変更禁止期間 (ユーザーによるパスワードの変更が許可されるまでに必要な時間) が定義された範囲内であることを確認します。

パスワードの変更禁止期間を設定することにより、最大パスワード履歴を上書きしようとしてパスワードを何回も変更するという行為を禁止することができます。パスワード変更要件を回避して元のパスワードに戻ろうとして、こういった行為を行うユーザーがいます。

オペレーティング システム	実装
Linux	変数 PASS_MIN_DAYS が **/etc/login.defs** に定義されています。
Solaris	**/etc/default/passwd** 内の変数 MINWEEKS は、パスワードの変更禁止期間 (週単位) を定義します。
HP-UX	この値は、**/etc/default/security** 内の変数 PASSWORD_MINDAYS によって制御されます。
Mac OS X	このオプションはサポートされていません。

例:

```
 <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>
```

ルート アクセス
root_login_from_console

使用法
`<item>` ` name: "root_login_from_console"` ` description: "This check makes sure that root can only log in from the system console (not remotely)."` `</item>`

このビルトイン関数は、"root" ユーザーが物理コンソールからリモート システムに直接ログインできるかどうかを確認します。

このチェックにより、アクセスから特定のユーザーをたどることのできる root アカウントの直接使用を禁止する管理上のベスト プラクティスを確認することができます。この場合、一般ユーザー アカウント (BSD システムの wheel グループのメンバー) を使用し、次に "**su**" (または **sudo**) を使用して、管理タスクを実行するための特権昇格を行います。

オペレーティング システム	実装
Linux と HP-UX	**/etc/securetty** が存在し、ここに "console" のみが含まれていることを確認します。
Solaris	**/etc/default/login** に行 "CONSOLE=/dev/console" が含まれていることを確認します。
Mac OS X	このオプションはサポートされていません。

パーミッション管理
accounts_bad_home_permissions

使用法
`<item>` ` name: "accounts_bad_home_permissions"` ` description: "This check reports user accounts that have home directories with incorrect user or group ownerships."` `</item>`

このビルトイン関数は、権限のない各ユーザーの home ディレクトリがそのユーザーに所属し、サード パーティ ユーザー (同じグループまたは "everyone" に所属する) がそのディレクトリに書き込めないことを確認します。一般的には、ユーザーの home ディレクトリは mode 0755 以上 (0700 など) に設定することをお勧めします。このチェックは、各 home ディレクトリが適切に構成されていれば合格となり、そうでない場合は不合格になります。ここでは、キーワード **mode** または **mask** を使用して、home ディレクトリのパーミッション レベルを指定します。**mode** キーワードは、指定レベルに正確に一致する home ディレクトリを受け入れ、**mask** キーワードは、指定レベル以上の home ディレクトリを受け入れます。

サード パーティが、あるユーザーの home ディレクトリに書き込みを行える場合、**~/.profile**、**~/.cshrc**、**~/.bashrc** ファイルを改ざんして、そのユーザーに任意のコマンドを実行させることが可能になります。

ファイルを同じグループのユーザー間で共有する必要がある場合、通常は、ユーザーの home ディレクトリではなく、そのグループから書き込める専用ディレクトリを用意することをお勧めします。

構成に問題がある home ディレクトリについては、"**chmod 0755 <ユーザー ディレクトリ>**" を実行して、オーナー権限を変更してください。

accounts_bad_home_group_permissions

使用法

```
<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>
```

このビルトイン関数は、**accounts_bad_home_permissions** と似ていますが、ユーザーの home ディレクトリがユーザーのプライマリ グループによりグループ所有されていることを確認します。

accounts_without_home_dir

使用法

```
<item>
 name: "accounts_without_home_dir"
 description: "This check reports user accounts that do not have home directories."
</item>
```

このビルトイン関数は、すべてのユーザーに home ディレクトリがあることを確認します。有効なディレクトリが各ユーザーに設定されている場合、合格となり、そうでない場合、不合格になります。この関数では、home ディレクトリのオーナー権限またはパーミッションはチェックされません。

通常は、システム上の各ユーザーに home ディレクトリが設定されていることをお勧めします。それは、ここからデータを読み取ったり、ここに書き込んだりするツールがあるためです (たとえば、**sendmail** は **~/.forward** ファイルをチェックします)。ユーザーがログインする必要がない場合、存在しないシェル (**/bin/false** など) を定義する必要があります。多くのシステム上で、home ディレクトリを持たないユーザーにも、ログイン権限が付与されますが、有効な home ディレクトリは / になります。

invalid_login_shells

使用法

```
<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>
```

このビルトイン関数は、各ユーザーに有効なシェルが **/etc/shells** に定義されていることを確認します。

/etc/shells ファイルは、Sendmail サーバーや FTP サーバーなどのアプリケーションによって、シェルがシステム上で有効であるかどうかの判断に使用されます。. このファイルがログイン プログラムによって使用されていない間に、管理者はこのファイルを使用して、システム上で有効なシェルを定義できます。**invalid_login_shells** チェックは、**/etc/shells** ファイルに定義されているように、**/etc/passwd** ファイル内の全ユーザーに有効なシェルが構成されていることを確認します。

ユーザーにパスワードを変更させるシェルとして **/sbin/passwd** を使用するなど、許可されない行為を禁止できます。ユーザーにログインを許可しない場合、無効なシェルを **/etc/shells** ("**/nonexistent**" など) に作成し、これを目的のユーザーに設定します。

有効なシェルを持たないユーザーがいる場合、そのユーザーに有効なシェルを定義してください。

login_shells_with_suid

使用法

```
<item>
 name: "login_shells_with_suid"
 description: "This check reports user accounts with login shells that have setuid or
setgid privileges."
</item>
```

このビルトイン関数は、"set-uid" 機能を持つシェルがないことを確認します。

"setuid" シェルとは、このシェルが起動すると、プロセスそのものが、自分でパーミッションを設定できるシェルのことです (たとえば、setuid "root" シェルは、スーパーユーザー特権を誰にでも付与できます)。

"setuid" シェルがあると、UID や GID の目的に反し、アクセス制御がはるかに複雑になります。

"setuid" である各シェルから SUID ビットを削除してください。

login_shells_writeable

使用法

```
<item>
 name: "login_shells_writeable"
 description: "This check reports user accounts with login shells that have group or
world write permissions."
</item>
```

このビルトイン関数は、ワールド/グループ書き込み可能なシェルがないことを確認します。

ワールド (またはグループ) 書き込み可能なシェルがあった場合、権限のないユーザーでもこのシェルをどのプログラムにでも置換できます。これにより、悪意あるユーザーが、そのシェルの他のユーザーにログイン時、任意のコマンドを実行するよう促すことが可能になります。

各シェルのパーミッションが適切に設定されていることを確認してください。

login_shells_bad_owner

使用法

```
<item>
 name: "login_shells_bad_owner"
 description: "This check reports user accounts with login shells that are not owned by
root or bin."
</item>
```

このビルトイン関数により、すべてのシェルが "root" ユーザーまたは "bin" ユーザーに所属することを確認します。

無効なパーミッションが設定されたシェルについては、あるユーザー所有のシェルを他のユーザーが使用できた場合、サードパーティユーザーがログインすると任意のコマンドが実行されるようにシェルを操作することが可能になります。

システムレベルのバイナリを変更できるのは "root" または "bin" のみにしておく必要があります。

パスワード ファイル管理

passwd_file_consistency

使用法

```
<item>
 name: "passwd_file_consistency"
 description: "This check makes sure /etc/passwd is valid."
</item>
```

このビルトイン関数は、**/etc/passwd** 内の各行が有効なフォーマットである (たとえば、7 つのフィールドがコロンで区切られている) ことを確認します。フォーマットが間違っている行があれば、レポートされ、チェックは不合格になります。

フォーマットが間違っている **/etc/passwd** ファイルがあると、複数のユーザー管理ツールの中断につながります。また、この症状は、侵入の原因や、カスタム ユーザー管理アプリケーション内のバグ、無効な名前のユーザーを追加しようとした者がいた (以前は、"toor:0:0" という名前のユーザーを作成して、root 特権を取得する手法が人気でした)、などを示しています。

不合格だった場合、管理者は、**/etc/passwd** から問題の行を削除するか修正することが必要です。

passwd_zero_uid

使用法

```
<item>
 name: "passwd_zero_uid"
 description: "This check makes sure that only ONE account has a uid of 0."
</item>
```

このビルトイン関数は、**/etc/passwd** 内に UID が "0" のアカウントが 1 つしかないことを確認します。これは、"root" アカウント用に予約されているアカウントですが、同じ特権のアクセス権を持つ UID 0 のアカウントを追加することは可能です。UID 0 のアカウントが 1 つだけの場合は合格で、それ以外の場合は不合格になります。

UID 0 はシステム上の root 特権を付与します。root ユーザーは、システム上で何でもできます。通常、これには、他のプロセス (またはカーネル) のメモリのスヌーピング、システム上の任意のファイルの読み取り、書き込みが含まれます。このアカウントは非常に強力なので、その使用は最低限に抑え、適切にセキュリティで保護する必要があります。

適切に管理を行う場合、各 UID ("一意" の識別子) は一意であることが求められます。"root" 特権を持つアカウントが 2 つ以上ある場合、1 人のシステム管理者がシステムに対して持つアカウンタビリティが否定されることになります。さらに、多くのシステムが、管理利用を追跡できるように、root の直接ログインをコンソールのみに制限しています。通常、システム管理者は、自身のアカウントに最初にログインして、**su** コマンドを使用して root になる方法を取ります。UID 0 アカウントの追加は、この制限に違反します。

"root" アクセスをユーザー間で共有する必要がある場合、**sudo** や **calife** などのツール (または Solaris の RBAC) を使用してください。UID が "0" のアカウントは 1 つしか存在できません。

passwd_duplicate_uid

使用法

```
<item>
 name: "passwd_duplicate_uid"
 description: "This check makes sure that every UID in /etc/passwd is unique."
</item>
```

このビルトイン関数は、**/etc/passwd** 内のすべてのアカウントに一意の UID があることを確認します。すべての UID が一意であれば合格になり、そうでない場合、不合格になります。

Unix システム上の各ユーザーは、そのユーザー ID (UID) の 0 ～ 65535 の数値で識別されます。2 人のユーザーが同じ UID を共有する場合、同じ特権が付与されるだけでなく、システム側は、この 2 人を同じユーザーと見なします。この状況は、どのアクションがどちらのユーザーで実行されたのかわからないため、アカウンタビリティ違反になります。通常、システムは、UID の逆引きを行い、ログ表示の際、その UID を共有する最初のアカウント名が使用されます。

CIS ベンチマークなどのセキュリティ基準では、ユーザー間での UID の共有は禁止されます。ユーザー間でファイルを共有する必要がある場合は、UID の共有ではなく、グループを使用してください。

システム上の各ユーザーには、一意の ID を付与する必要があります。

passwd_duplicate_gid

使用法

```
<item>
 name: "passwd_duplicate_gid"
 description: "This check makes sure that every GID in /etc/passwd is unique."
</item>
```

このビルトイン関数は、各ユーザーのプライマリ グループ ID (GID) が一意であることを確認します。すべてのユーザーに一意の GID がある場合は合格になり、そうでない場合は不合格になります。

セキュリティ上は、ユーザーごとにグループを作成することが推奨されます (通常は、ユーザー名と同じ名前が付きます)。このセットアップにより、そのユーザーにより作成されたファイルは、そのプライマリ グループに所属し、そのユーザーにしか変更できないため、"デフォルトでセキュリティ保護されている" と見なされます。そのユーザーが、そのファイルをグループの他のメンバーの所有としたい場合は、元の所有者であるユーザーが明示的に **chgrp** コマンドを実行して、オーナー権限を変更する必要があります。

この手法のもう一つの利点は、グループ メンバーシップ管理を、**/etc/passwd** と **/etc/group** の混合ではなく、1 つのファイル (**/etc/group**) に統一できることです。

各ユーザーに対して、同じ名前のグループを作成してください。グループのオーナー権限は、**/etc/group** からのみ管理します。

passwd_duplicate_username

使用法

```
<item>
 name: "passwd_duplicate_username"
 description: "This check makes sure that every username in /etc/passwd is unique."
```

```
</item>
```

このビルトイン関数は、**/etc/passwd** 内の各ユーザー名が一意であることを確認します。各ユーザー名が一意であれば合格で、そうでない場合に不合格になります。

/etc/passwd 内に重複するユーザー名があると、どのアカウントの特権が使用されているのかわからないため、問題が生じます。

adduser コマンドでは、重複するユーザー名は作成されません。重複するユーザー名がある場合、通常は、システムのセキュリティが侵害されている、ユーザー管理ツールがおかしくなっている、または **/etc/passwd** ファイルが手動で編集されていることを示しています。

重複するユーザー名は削除するか、別のユーザー名を設定してください。

passwd_duplicate_home

使用法

```
<item>
 name: "passwd_duplicate_home"
 description: "(arbitrary user comment)"
</item>
```

このビルトイン関数は、**/etc/passwd** 内の非システム ユーザー (UID が 101 以上) にそれぞれ一意の home ディレクトリがあることを確認します。

/etc/passwd 内の各ユーザー名が、一意の home ディレクトリを持つ必要があります。ユーザー間で同じ home ディレクトリを共有する場合、スタートアップ ファイル (**.profile** など) を変更したり、home ディレクトリに不正バイナリを混入することにより、他のユーザーに任意のコマンドを実行させることが可能になります。さらに、共有 home ディレクトリは、ユーザー アカウンタビリティ違反になります。

コンプライアンス要件は、各ユーザーが一意の home ディレクトリを持っていることを必須とします。

passwd_shadowed

使用法

```
<item>
 name: "passwd_shadowed"
 description: "(arbitrary user comment)"
</item>
```

このビルトイン チェックは、**/etc/passwd** 内のすべてのパスワードが "シャドウ" になっている (つまり、別のファイルに格納されている) ことを確認します。

/etc/passwd はワールドから読み取り可能であるため、ユーザーのパスワード ハッシュをここに格納した場合、ここにアクセスできるユーザーであれば誰でも、そのパスワード ハッシュに対してパスワード クラッキング プログラムを実行できることになります。ユーザーのパスワードを総当たり攻撃 (パスワードを変えてログインを繰り返す) によって当てようとする試みは、通常、システム ログ ファイルから検出できます。**/etc/passwd** ファイルにパスワード ハッシュが含まれている場合、そのファイルをオフラインにコピーできてしまえば、パスワード クラッキング プログラムでハッシュを解読することが可能になります。この方法では、検出されることなくユーザー パスワードを取得することができます。

大部分の新しい Unix システムでは、シャドウ パスワード ファイルが採用されています。システムでシャドウ パスワードを有効にする方法については、システムのマニュアルを参照してください。

passwd_invalid_gid

使用法

```
<item>
 name: "passwd_invalid_gid"
 description: "This check makes sure that every GID defined in /etc/passwd exists in
/etc/group."
</item>
```

このビルトイン関数は、**/etc/passwd** 内の各グループ ID (GID) が **/etc/group** に存在していることを確認します。各 GID が適切に定義されている場合は合格になり、そうでない場合は不合格になります。

グループ ID が **/etc/passwd** に定義されるたびに、即時 **/etc/group** にも設定する必要があります。この処置を怠ると、システムの一貫性が損なわれ、問題の原因になる場合があります。

たとえば、ユーザー ("bob") が UID 1000 と GID 4000 を持っている場合、GID が **/etc/group** に定義されていなければ、このユーザーのプライマリ グループは、このユーザーにいかなる特権も付与できないことになります。数か月後に、システム管理者が **/etc/group** を編集して、グループ "admin" を追加し、"未使用" の GID 4000 をこのグループに設定した場合、ユーザー "bob" は意図せず、デフォルトで "admin" グループに所属することになります。

/etc/group を編集して、不足している GID を追加してください。

グループ ファイル管理

group_file_consistency

使用法

```
<item>
 name: "group_file_consistency"
 description: "This check makes sure /etc/group is valid."
</item>
```

このビルトイン関数は、**/etc/group** 内の各行が有効なフォーマットである (たとえば、3 つの項目がコロンで区切られ、ユーザーリストを含んでいる) ことを確認します。フォーマットが間違っている行があれば、レポートされ、チェックは不合格になります。

フォーマットが間違っている **/etc/group** ファイルがあると、複数のユーザー管理ツールの中断につながります。また、この症状は、侵入の原因や、カスタム ユーザー管理アプリケーション内のバグ、無効なグループ名のユーザーを追加しようとした者がいた、などを示しています。

/etc/group ファイルを編集して、行のフォーマットを修正してください。

group_zero_gid

使用法

```
<item>
 name: "group_zero_gid"
```

```
 description: "This check makes sure that only ONE group has a gid of 0."
</item>
```

このビルトイン関数は、グループ ID (GID) が 0 のグループが 1 つだけであることを確認します。GID が 0 のグループが 1 つだけの場合は合格で、そうではない場合は不合格になります。

GID "0" は、このグループのメンバーであるユーザーがメンバー root のプライマリ グループでもある、ということです。これは、root グループ パーミッションを持つ任意のファイルへの root 特権を付与することになります。

管理者グループを定義する場合は、"admin" グループを作成してください。

group_duplicate_name

使用法

```
<item>
 name: "group_duplicate_name"
 description: "This check makes sure that every group name in /etc/group is unique."
</item>
```

このビルトイン チェックは、各グループ名が一意であることを確認します。各ユーザー名が一意であれば合格で、そうでない場合に不合格になります。

/etc/group 内に重複するグループ名があると、どのグループの特権が使用されているのかわからないため、問題が生じます。重複するグループ名があれば、もともとは設定するべきではなかったメンバーや特権が設定されてしまうようなことが起こります。

重複グループ名は削除するか、名前を変更してください。

group_duplicate_gid

使用法

```
<item>
 name: "group_duplicate_gid"
 description: "(arbitrary user comment)"
</item>
```

Unix システム上の各グループは、そのグループ ID (GID) の 0 ～ 65535 の数値で識別されます。2 つのグループが同じ GID を共有する場合、同じ特権が付与されるだけでなく、システム側は、この 2 つを同じグループと見なします。これは、ユーザー特権を分離するというグループの目的から外れるものになります。

セキュリティ上、グループ間での GID の共有は禁止されています。2 つのグループに同じ特権を持たす必要がある場合は、そのグループは同じユーザーで構成されていることが必要になります。

重複するグループは削除するか、一方に新しい一意の GID を割り当ててください。

group_duplicate_members

使用法

```
<item>
 name: "group_duplicate_members"
 description: "This check makes sure that every member of a group is listed once."
</item>
```

このビルトイン関数は、グループの各メンバーがグループ リストに 1 回しか出現していないことを確認します。各メンバーが一意であれば合格で、そうでない場合は不合格になります。

グループの各メンバーはグループ リストに 1 回のみの出現である必要があります。同じメンバーが複数回リストに出現しても、基のオペレーティング システムに問題は発生しませんが、システム管理者にとって、特権の取り消し作業が複雑になります。たとえば、グループ "admin" のメンバー構成が "alice,bob,charles,daniel,bob" である場合、"bob" の特権を取り消そうと思えば、"bob" を 2 回削除することが必要になります。

各メンバーが、リストに 1 回のみ出現していることを確認します。

group_nonexistant_users

使用法

```
<item>
 name: "group_nonexistant_users"
 description: "This check makes sure that every member of a group actually exists."
</item>
```

このチェックは、グループ内の各メンバーが **/etc/passwd** 内に存在していることを確認します。

/etc/group に存在していないユーザーがいることは、管理上好ましくありません。そのユーザーは、入力ミスされているか、システムから削除されているのにグループから削除されていないかのどちらかになります。

"ゴースト" ユーザーが **/etc/group** に含まれていることは推奨されません。同じユーザー名のユーザーが後に追加された場合、そのユーザーに、付与すべきではないグループ特権が付与される可能性が生じます。

存在していないユーザーは **/etc/group** から削除してください。

ルート環境

dot_in_root_path_variable

使用法

```
<item>
 name: "dot_in_root_path_variable"
 description: "This check makes sure that root's $PATH variable does not contain any
relative path."
</item>
```

このチェックは、現在の作業ディレクトリ (".") が、root ユーザーの実行可能パスに含まれていないことを確認します。このことを保証することにより、悪意あるユーザーが、現在の作業ディレクトリにインストールできるトロイの木馬を、root としてログインしている管理者に実行させることにより、自分をスーパーユーザーに昇格させる手法を禁止することができます。

writeable_dirs_in_root_path_variable

使用法

```
<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>
```

このチェックは、root ユーザーの PATH 変数にある、すべてのワールド/グループ書き込み可能ディレクトリをレポートします。このチェックにより返されるディレクトリはすべて、よく確認して、ディレクトリへの不要なワールド/グループ書き込み可能パーミッションがあれば、次のコマンドで削除してください。

```
 # chmod go-w path/to/directory
```

ファイル パーミッション
find_orphan_files

使用法

```
<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>
```

このチェックは、システム上にオーナーがいないすべてのファイルをレポートします。

デフォルトでは、検索は "/" ディレクトリの下で再帰的に実行されます。このことにより、このチェックは、リモート システム上に存在するファイル数に応じて、実行時間が非常に長くなる場合があります。ただし、必要に応じて、オプション キーワード **basedir** を使用して、検索対象のデフォルトの基本ディレクトリを変更できます。また、別のオプション キーワード **ignore** を使用して、基本ディレクトリ内から特定のファイルを検索対象から外すこともできます。デフォルトのファイル システムの検索では、オプション キーワード **dir** で指定されていない限り、NFS にマウントされているディレクトリはすべて無視されます。

検索の性質上、スキャンされるシステムのタイプに応じて、数時間かかるのが通常です。デフォルトでは、Nessus がこのチェックの結果処理を停止するタイムアウト値は 5 時間に設定されており、この値は変更できません。

例:

```
 <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

使用法

```
<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>
```

このチェックは、リモート システム上のワールド書き込み可能なすべてのファイルをレポートします。リモート システム上にはワールド書き込み可能ファイルはない方が理想的です。この場合、このチェック結果として、何も表示されません。ただし、組織上の要件によっては、ワールド書き込み可能ファイルが必要になることもあります。このチェックが返される項目はすべて慎重に監査して、ワールド書き込み可能属性が必ずしも必要でないファイルについては、次のコマンドで削除してください。

```
 # chmod o-w world_writeable_file
```

デフォルトでは、検索は "/" ディレクトリの下で再帰的に実行されます。このことにより、このチェックは、リモート システム上に存在するファイル数に応じて、実行時間が非常に長くなる場合があります。ただし、必要に応じて、オプション キーワード **basedir** を使用して、検索対象のデフォルトの基本ディレクトリを変更できます。また、別のオプション キーワード **ignore** を使用して、基本ディレクトリ内から特定のファイルを検索対象から外すこともできます。デフォルトのファイル システムの検索では、オプション キーワード **dir** で指定されていない限り、NFS にマウントされているディレクトリはすべて無視されます。

検索の性質上、スキャンされるシステムのタイプに応じて、数時間かかるのが通常です。デフォルトでは、Nessus がこのチェックの結果処理を停止するタイムアウト値は 5 時間に設定されており、この値は変更できません。

例:

```
 <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

使用法

```
<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>
```

このチェックは、ワールド書き込み可能であるが、スティッキー ビットが設定されていない、リモート システム上のすべてのディレクトリをレポートします。スティッキー ビットがすべてのワールド書き込み可能ディレクトリに設定されていることを確認することにより、ディレクトリ内のファイルのオーナーのみそのファイルを削除できることを保証できます。これにより、オーナー以外のユーザーによりファイルが偶発的または意図的に削除されないようにできます。

デフォルトでは、検索は "/" ディレクトリの下で再帰的に実行されるので、このチェックは、リモート システム上に存在するファイル数に応じて、実行時間が非常に長くなる場合があります。ただし、必要に応じて、オプション キーワード **basedir** を使用して、検索対象のデフォルトの基本ディレクトリを変更できます。また、別のオプション キーワード **ignore** を使用して、基本ディレクトリ内から特定のファイルを検索対象から外すこともできます。デフォルトのファイル システムの検索では、オプション キーワード **dir** で指定されていない限り、NFS にマウントされているディレクトリはすべて無視されます。

検索の性質上、スキャンされるシステムのタイプに応じて、数時間かかるのが通常です。デフォルトでは、Nessus がこのチェックの結果処理を停止するタイムアウト値は 5 時間に設定されており、この値は変更できません。

例:

```
 <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

使用法

```
<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>
```

このチェックは、ワールド読み取り可能であるすべてのファイルをレポートします。たとえば、ユーザーの home ディレクトリの読み取り可能ファイルをチェックすることにより、他のユーザーからアクセスできる機密ファイル (プライベート SSH キーなど) がないことを保証できます。

デフォルトでは、検索は "/" ディレクトリの下で再帰的に実行されるので、このチェックは、リモート システム上に存在するファイル数に応じて、実行時間が非常に長くなる場合があります。ただし、必要に応じて、オプション キーワード **basedir** を使用して、検索対象のデフォルトの基本ディレクトリを変更できます。また、別のオプション キーワード **ignore** を使用して、基本ディレクトリ内から特定のファイルを検索対象から外すこともできます。デフォルトのファイル システムの検索では、オプション キーワード **dir** で指定されていない限り、NFS にマウントされているディレクトリはすべて無視されます。

検索の性質上、スキャンされるシステムのタイプに応じて、数時間かかるのが通常です。デフォルトでは、Nessus がこのチェックの結果処理を停止するタイムアウト値は 5 時間に設定されており、この値は変更できません。

例:

```
 <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

使用法

```
<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>
```

このチェックは、SUID/SGID ビットが設定されているすべてのファイルをレポートします。このチェックによってレポートされているファイルはすべて、注意して監査する必要があります。特に、シェル スクリプト、社内開発の実行可能ファイル (システムとともに出荷されない実行可能ファイルなど) は注意を要します。SUID/SGID ファイルには、標準ユーザーの特権を、ファイルのオーナーまたはグループの特権に昇格できるというリスクがあります。このようなファイル/スクリプトがどうしても必要な場合は、昇格特権でファイルを作成する許可が与えられていないかどうか、注意深く検証する必要があります。

デフォルトでは、検索は "/" ディレクトリの下で再帰的に実行されます。このことにより、このチェックは、リモート システム上に存在するファイル数に応じて、実行時間が非常に長くなる場合があります。ただし、必要に応じて、オプション キーワード **basedir** を使用して、検索対象のデフォルトの基本ディレクトリを変更できます。また、別のオプション キーワード **ignore** を使用して、基本ディレクトリ内から特定のファイルを検索対象から外すこともできます。デフォルトのファイル システムの検索では、オプション キーワード **dir** で指定されていない限り、NFS にマウントされているディレクトリはすべて無視されます。

検索の性質上、スキャンされるシステムのタイプに応じて、数時間かかるのが通常です。デフォルトでは、Nessus がこのチェックの結果処理を停止するタイムアウト値は 5 時間に設定されており、この値は変更できません。

例:

```
<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
このビルトイン チェックは、ユーザーの home ディレクトリ内のローカライズ ファイルがそのユーザーまたは root によって所有されていることを確認します。

1 つ以上のファイルを、"file" トークンを使用してリストアップします。ただし、"file" トークンがない場合、デフォルトで、次のファイルが検索されます。

- **.login**
- **.cschrc**
- **.logout**
- **.profile**
- **.bash_profile**
- **.bashrc**
- **.bash_logout**
- **.env**
- **.dtprofile**
- **.dispatch**
- **.emacs**
- **.exrc**

例:

```
<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
このビルトイン チェックは、ユーザーの home ディレクトリ内のローカライズ ファイルがそのユーザーのプライマリ グループまたは root によってグループ所有されていることを確認します。

1 つ以上のファイルを、"file" トークンを使用してリストアップします。ただし、"file" トークンがない場合、デフォルトで、次のファイルが検索されます。

- **.login**
- **.cschrc**

- **.logout**
- **.profile**
- **.bash_profile**
- **.bashrc**
- **.bash_logout**
- **.env**
- **.dtprofile**
- **.dispatch**
- **.emacs**
- **.exrc**

例:

```
 <item>
  name: "home_dir_localization_files_group_check"
  description: "Check file .foo/.foo2"
  file: ".foo"
  file: ".foo2"
  file: ".foo3"
 </item>
```

疑わしいファイル コンテンツ
admin_accounts_in_ftpusers

使用法

```
<item>
 name: "admin_accounts_in_ftpusers"
 description: "This check makes sure every account whose UID is below 500 is present in
/etc/ftpusers."
</item>
```

このチェックは、すべての admin アカウント、つまり 500 未満の UID のユーザーが **/etc/ftpusers**、**/etc/ftpd/ftpusers**、または **/etc/vsftpd.ftpusers** に存在していることを確認します。

不要ファイル
find_pre-CIS_files

使用法

```
<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>
```

このチェックは、Red Hat CIS ベンチマーク認定に合格するための Center for Internet Security (CIS) 要件用のチェックです。このチェックは、CIS Red Hat ベンチマークに基づいて Red Hat システムの構成/強化を行ったユーザーにとって特に有益です。CIS ベンチマーク ツールは、システム強化プロセス中に変更された可能性のあるすべてのシステム ファイルをバックアップするバ

ックアップ スクリプトを提供します。これらのファイルはキーワード "**-preCIS**" が接尾辞になっています。すべてのベンチマーク推奨が適用され、システムが稼働状況に戻ったところで、これらのファイルは削除されます。このチェックは、リモート システム上に "**preCIS**" ファイルが存在していないことを確認します。

デフォルトでは、検索は "/" ディレクトリの下で再帰的に実行されるので、このチェックは、リモート システム上に存在するファイル数に応じて、実行時間が非常に長くなる場合があります。ただし、必要に応じて、オプション キーワード **basedir** を使用して、検索対象のデフォルトの基本ディレクトリを変更できます。また、別のオプション キーワード **ignore** を使用して、基本ディレクトリ内から特定のファイルを検索対象から外すこともできます。

検索の性質上、スキャンされるシステムのタイプに応じて、数時間かかるのが通常です。デフォルトでは、Nessus がこのチェックの結果処理を停止するタイムアウト値は 5 時間に設定されており、この値は変更できません。

条件

Unix ポリシー内に **if/then/else** 論理を定義することができます。条件により、1 つのファイルで複数の構成を処理できるようになります。たとえば、同じポリシー ファイルで、**if/then/else** シンタックスを適切に使用して、Postfix 設定と Sendmail 設定のチェックを行うことができます。

条件を実行するシンタックスは次のとおりです。

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

例:

```
<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>
```

"サイレント" チェックなので、条件の失敗または成功はレポートに表示されません。

条件タイプは "**and**" または "**or**" のいずれかになります。

NetApp Data ONTAP

ここでは、NetApp Data ONTAP コンプライアンス チェックのフォーマットと関数、および各設定の論理的根拠について説明します。

引用符の用法:

監査フィールドを囲む一重引用符と二重引用符は、次の 2 つの場合を除いてどちらでも使用可能です。

1. CRLF など、文字をリテラルに解釈する必要がある特殊フィールドを含む Windows コンプライアンス チェックでは、一重引用符を使用します。文字列として解釈する組み込みフィールドはエスケープする必要があります。

たとえば、次のとおりです。

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

2. 二重引用符は WindowsFiles の "include_paths" と "exclude_paths" を使用する際に必要です。

どのフィールドタイプ (description、value_data、regex など) においても、一重引用符または二重引用符を含む文字列を使用する場合は、次のいずれかの方法を用います。

a. 外側の引用符とは異なる引用符タイプを使用する。

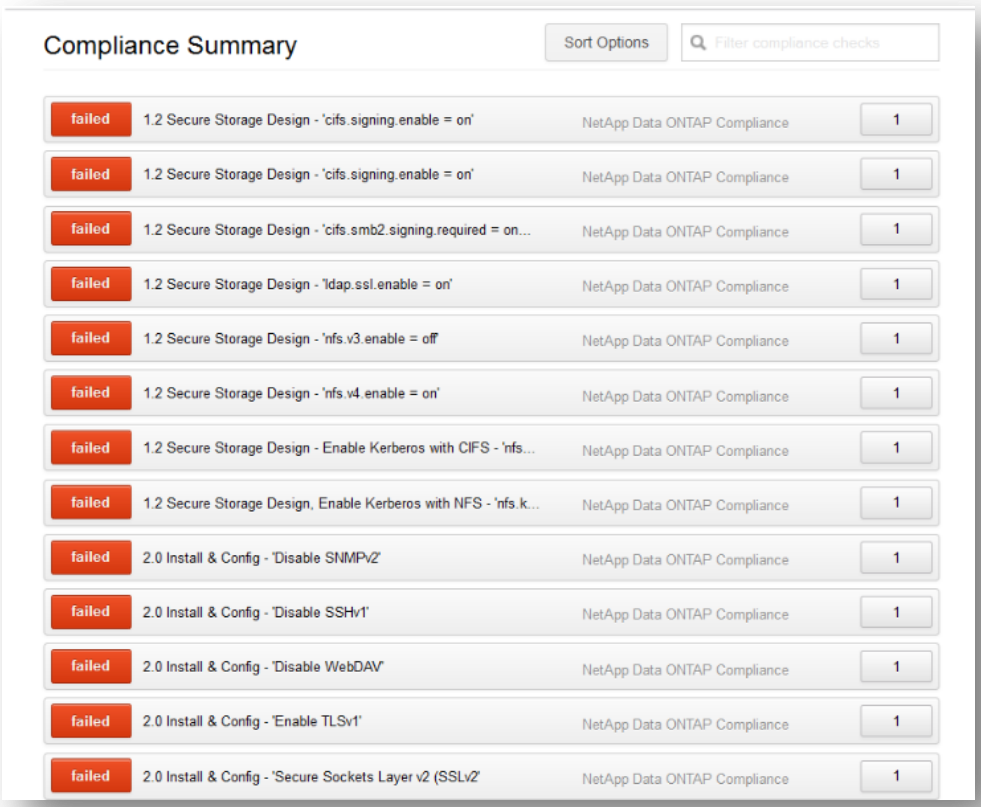
例:

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

b. 組み込み引用符はバックスラッシュでエスケープする (二重引用符のみ)。

例:

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



必要なユーザー特権

NetApp Data ONTAP システムに対してコンプライアンス スキャンを正しく実行するには、認証ユーザーが以下の特権を持っていなければなりません。

NetApp Data ONTAP フィルタ用の `root` 資格情報

上記の特権に加えて、NetApp Data ONTAP コンプライアンス チェック用の監査ポリシーと Nessus Plugin ID 66934 (NetApp Data ONTAP Compliance Checks) が必要です。

デバイスに対するスキャンを実行するには、まずは監査ポリシーを作成します。次に、ポリシーの "**Credentials**" タブの "**SSH settings**" メニューを選択して、`root` 資格情報を入力します。ポリシーの "**Plugins**" タブで、"Policy Compliance" プラグイン ファミリーを選択し、"**NetApp Data ONTAP Compliance Checks**" というタイトルのプラグイン ID 66934 を有効にします。次に、"**Preferences**" タブから "NetApp Data ONTAP Compliance Checks" ドロップダウンを選択し、Tenable サポート ポータルから NetApp **`.audit`** ファイルを追加します。最後に、ポリシーを保存して、スキャンを実行します。

`root` 資格情報の入力がオプションとして含まれない場合、権限が低いアカウントを作成して、監査を実行できます。

1. 新しいロールを作成する (`nessus_audit` など):

   ```
   # role add nessus_audit -a login-ssh,cli-version,cli-options,cli-uptime
   ```

2. ロールをグループに割り当てる (`nessus_admins` など):

   ```
   # group add nessus_admins -r nessus_audit
   ```

3. グループをユーザーに割り当てる:

   ```
   # useradmin user add nessus -g nessus_admins
   ```

チェック タイプ:CONFIG_CHECK

Check Point コンプライアンス チェックは、**custom_item** カプセルで囲まれ、CONFIG_CHECK が指定されているものになります。これは、他の **.audit** ファイルと同じように処理され、NetApp Data ONTAP システムを実行するシステムに対して使用されます。CONFIG_CHECK チェックは、2 つ以上のキーワードで構成されます。キーワード **type** と **description** は必須です。この後に 1 つ以上のキーワードが続きます。チェックは、"options" コマンド出力を監査することにより行われます。

キーワード

次の表に、NetApp Data ONTAP コンプライアンス チェックにおける各キーワードの使用方法を示します。

キーワード	使用例とサポートされている設定
type	"CHECK_CONFIG" は、NetApp Data ONTAP "show configuration" 出力に指定の構成項目が存在していることを確認します。
description	"**description**" キーワードは、実行するチェックの簡単な説明を追加します。**description** フィールドには一意のテキストを指定して、**description** フィールドが重複するチェックは作成しないことを強く推奨します。Tenable の SecurityCenter は、一意のプラグイン ID 番号を **description** フィールドに基づいて自動的に生成します。 例: `description: "1.0 Require strong Password Controls - 'min-password-length >= 8'"`
info	"**info**" キーワードは、実行するチェックに関する詳細な説明を追加するのに使用します。チェックの論理的根拠として、規則であったり、詳細情報が記載されている URL であったり、企業ポリシーなどを指定できます。複数の **info** フィールドを、テキストをパラグラフとしてフォーマットするために 1 行に 1 つずつ追加することができます。使用できる **info** フィールド数に制限はありません。 各 "**info**" タグは、改行なしで別の行に入力する必要があります。複数行が必要な場合は (フォーマット上の理由などにより)、"**info**" タグを追加します。 例: `info: "Enable palindrome-check on passwords"`
severity	"**severity**" キーワードは、実行するチェックの重大度を指定します。 例: `severity: MEDIUM` 重大度は HIGH、MEDIUM、または LOW に設定できます。
regex	"**regex**" キーワードにより、構成項目設定の検索で、特定の正規表現への照合を行うことができます。 例: `regex: "set snmp .+"` 次のメタ文字には特別な処理が必要です:+ \ * ( ) ^ これらの文字は、リテラルで解釈する場合は、2 つのバックスラッシュでエスケープするか、

	角括弧 "[]" で括ります。. ? " ' などの他の文字は、リテラルで解釈するには、バックスラッシュを 1 つだけ使用します。 これは、コンパイラがこれらの文字を処理する方法に関係します。 チェックに "regex" タグが設定され、"expect"、"not_expect"、"number_of_lines" タグは設定されていない場合、チェックは、regex と一致する全行をレポートします。
expect	このキーワードにより、"**regex**" タグに一致する構成項目を監査できます。また、"**regex**" タグが使用されていない場合、構成全体から "**expect**" 文字列が検索されます。 "**regex**" によって検出された構成行が "**expect**" タグと一致している場合、チェックは合格になります。"**regex**" が設定されていない場合は、"**expect**" 文字列が構成から検出された場合、合格になります。 例: `regex: "set password-controls complexity"` `expect: "set password-controls complexity [1-4]"` 上記の場合、"**expect**" タグは、複雑度が 1 ～ 4 の値に設定されていることを確認します。
not_expect	このキーワードにより、構成内に含まれてはならない構成項目を検索できます。 これは "**expect**" の反対の動作になります。"**regex**" によって検出された構成行が "**not_expect**" タグと一致しない場合、チェックは合格になります。"**regex**" タグが設定されていない場合は、構成に "**not_expect**" 文字列が検出されなければ合格です。 例: `regex: "set password-controls password-expiration"` `not_expect: "set password-controls password-expiration never"` 上記の場合は、"**not_expect**" タグにより、password-controls が "never" に設定されていないことがチェックされます。

CONFIG_CHECK の例

NetApp Data ONTAP デバイスに対する CONFIG_CHECK の使用例を次に示します。

```
<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>
```

条件

NetApp Data ONTAP 監査ポリシー内に **if/then/else** 論理を定義することができます。条件により、1 つのファイルで複数の構成を処理できるようになります。

条件を実行するシンタックスは次のとおりです。

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

例：

```
<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>
```

条件はレポート内には表示されません。つまり、条件が失敗したか成功したかは表示されません ("サイレント" チェック)。

条件タイプは "**and**" または "**or**" のいずれかになります。

レポート

<then> または <else> 内で実行して、目的の PASSED/FAILED 条件を適用できます。

```
<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>
```

"report type" の有効値は PASSED、WARNING、および FAILED です。

IBM iSeries 構成監査コンプライアンス ファイル リファレンス

ここでは、IBM iSeries コンプライアンス チェックのフォーマットと関数、および各設定の論理的根拠について説明します。

引用符の用法:

監査フィールドを囲む一重引用符と二重引用符は、次の 2 つの場合を除いてどちらでも使用可能です。

1. CRLF など、文字をリテラルに解釈する必要がある特殊フィールドを含む Windows コンプライアンス チェックでは、一重引用符を使用します。文字列として解釈する組み込みフィールドはエスケープする必要があります。

たとえば、次のとおりです。

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

2. 二重引用符は WindowsFiles の "include_paths" と "exclude_paths" を使用する際に必要です。

どのフィールドタイプ (description、value_data、regex など) においても、一重引用符または二重引用符を含む文字列を使用する場合は、次のいずれかの方法を用います。

a. 外側の引用符とは異なる引用符タイプを使用する。

例:

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

b. 組み込み引用符はバックスラッシュでエスケープする (二重引用符のみ)。

例:

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

必要なユーザー特権

iSeries システムに対してコンプライアンス スキャンを正しく実行するには、認証ユーザーが以下の特権を持っていなければなりません。

1. (*ALLOBJ) または監査 (*AUDIT) 権限を持つユーザーは、すべてのシステム値を監査することができます。このようなユーザーは通常、(*SECOFR) クラスに属します。

2. (*USER) または (*SYSOPR) クラスのユーザーはほとんどの値を監査できますが、QAUDCTL、QAUDENDACN、QAUDFRCLVL、QAUDLVL、QAUDLVL2、および QCRTOBJAUD は監査できません。

ユーザーが値にアクセスする権限を持たない場合、*NOTAVL という値が返されます。

チェック タイプ

すべての IBM iSeries コンプライアンス チェックは、**check_type** カプセルで囲われ、"AS/400" 指定を含んでいる必要があります。これは、この **.audit** ファイルが、IBM iSeries システムを実行しているシステム用であることを識別するためのものです。

例:

```
<check_type:"AS/400">
```

他のコンプライアンス監査タイプとは異なり、追加のタイプやバージョンのキーワードはありません。

キーワード

次の表に、IBM iSeries コンプライアンス チェックにおける各キーワードの使用方法を示します。

キーワード	使用例とサポートされている設定
type	AUDIT_SYSTEMVAL SHOW_SYSTEMVAL
systemvalue	このキーワードは、IBM iSeries システム内でチェックする値を指定するのに使用されます。 例: systemvalue: "QALWUSRDMN"
description	このキーワードは、実行するチェックの簡単な説明を追加します。**description** フィールドには一意のテキストを指定して、description フィールドが重複するチェックは作成しないことを強く推奨します。Tenable の SecurityCenter は、一意のプラグイン ID 番号を **description** フィールドに基づいて自動的に生成します。 例: description: "Allow User Domain Objects (QALWUSRDMN) - '*all'"
value_type	このキーワードは、IBM iSeries システム上でチェックする値のタイプ ("POLICY_DWORD" または "POLICY_TEXT") を定義するのに使用されます。 例: value_type: "POLICY_DWORD" 例: value_type: "POLICY_TEXT"
value_data	このキーワードは、システム値として期待データ値を定義します。

	例: value_type: "^([6-9]|[1-9][0-9]+)$"
check_type	このキーワードは、データ値に対して使用するチェック タイプを定義します。 例: 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" 例: <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	このキーワードは、実行するチェックに関する詳細な説明を追加するのに使用します。このキーワードにより、規則、URL、企業ポリシー、または設定が必要なその他の理由を示します。複数の **info** フィールドを、テキストをパラグラフとしてフォーマットするために 1 行に 1 つずつ追加することができます。使用できる **info** フィールド数に制限はありません。 例: info: "\nref : http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/books/sc415302.pdf pg. 21"

カスタム項目

カスタム項目は、前記のキーワードに基づいて定義される 1 つの完全なチェックです。次に、使用可能なカスタム項目のタイプを示します。各チェックが "**<custom_item>**" タグで始まり、"**</custom_item>**" で終わります。これらのタグ内に、コンプライアンス チェック パーサーによって解釈され、チェックを実行する 1 つ以上のキーワードが配置されます。

カスタム監査チェックでは、終了タグとして "**</custom_item>**" と "**</item>**" のどちらでも使用できます。

AUDIT_SYSTEMVAL

"AUDIT_SYSTEMVALUE" は、"**systemvalue**" キーワードによって識別される構成設定値を監査します。監査対象値の比較タイプについては、"**check_type**" キーワードが指定します。

```
<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

"SHOW_SYSTEMVAL" 監査は、"**systemvalue**" キーワードにより識別される構成設定値をレポートするだけです。

```
<custom_item>
 type: SHOW_SYSTEMVAL
 systemvalue: "QAUDCTL"
 description: "show QAUDCTL value"
 severity: MEDIUM
</custom_item>
```

条件

IBM iSeries ポリシー内に **if/then/else** 論理を定義することができます。これにより、監査が合格した場合、合格/不合格の結果ではなく、警告メッセージを返すことができます。

条件を実行するシンタックスは次のとおりです。

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

例:

```
<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>
```

"サイレント" チェックなので、条件の失敗または成功はレポートに表示されません。

条件タイプは "**and**" または "**or**" のいずれかになります。

VMware vCenter/ESXi 構成監査コンプライアンス ファイル リファレンス

ここでは、VMware vCenter/ESXi コンプライアンス チェックのフォーマットと関数、および各設定の論理的根拠について説明します。

Nessus では、構成を抽出し、関連付けられている **.audit** ファイルにリストアップされているチェックに基づいて監査を実行することにより、ネイティブ API を介して VMware を監査できます。

要件

VMware システムに対してコンプライアンス スキャンを正しく実行するには、ユーザーが以下を有している必要があります。

1. VMware vCenter または ESXi 用管理者資格情報。Tenable では ESXi (ESX/ESXi 上で VM を管理するための無料のインタフェース) と vCenter (1 つ以上で ESX/ESXi サーバーを管理するための VMware から利用できる有料のアドオン製品) の両方に対応した API が開発されています。このプラグインは、ESXi または vCenter いずれかの資格情報を利用します。

2. VMware vCenter/ESXi コンプライアンス チェック用の監査ポリシー。

3. プラグイン ID 64455 (VMware vCenter/ESXi コンプライアンス チェック)

サポートされているバージョン

現在 Nessus は、ESXi 4.x と 5.x、および vCenter 4.x と 5 を監査できます。

チェック タイプ

VMware **.audit** 機能用のシンタックスは、この機能を実行するために XPATH および XSLT に大きく依存しています。

VMware 監査は、次の 3 つのチェック タイプをサポートしています。

AUDIT_VM

このチェック タイプでは、仮想マシン設定を監査できます (詳細については付録 C を参照してください)。

```
<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

このチェック タイプでは、ESX/ESXi サーバー設定を監査できます。

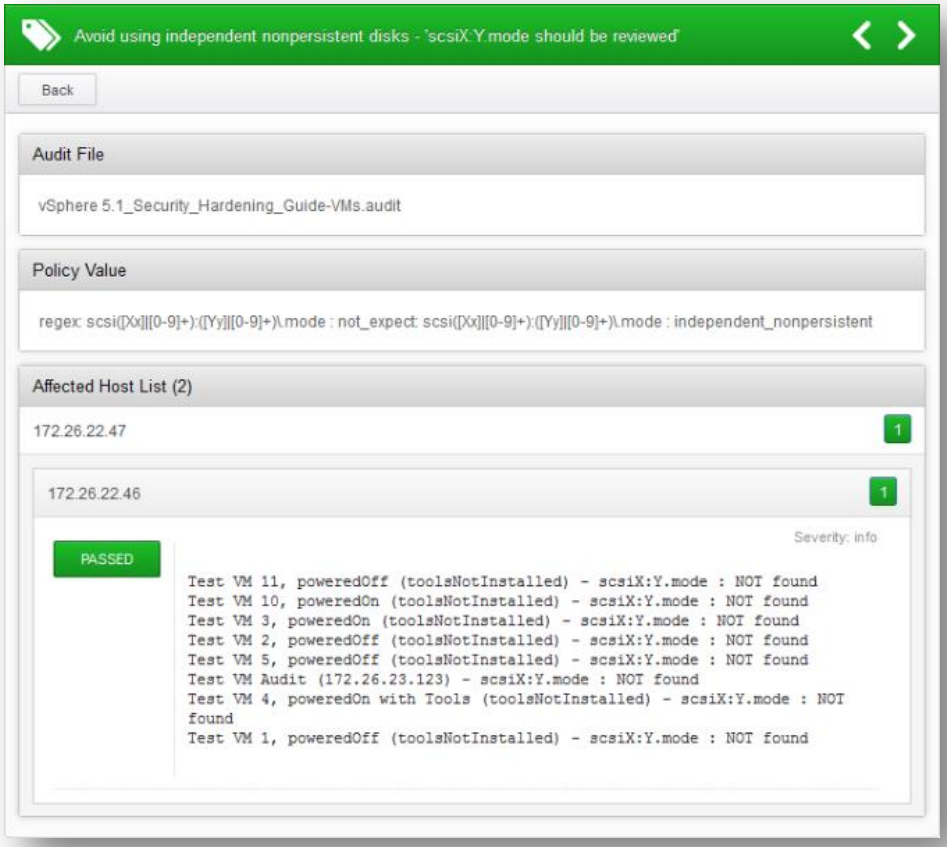
```
<custom_item>
 type: AUDIT_ESX
 description : "ESX/ESXi Setting - Syslog.global.logDir"
 xsl_stmt: "<xsl:template match=\"audit:returnval\">"
 xsl_stmt: "Syslog.global.logDir = <xsl:value-of
       select=\"audit:propSet/audit:val[@xsi:type='HostConfigInfo']/audit:option[audit
       :key[text()='Syslog.global.logDir']]/audit:value\"/>"
 xsl_stmt: "</xsl:template>"
 expect: "Syslog.global.logDir : /foo/bar"
</custom_item>
```

AUDIT_VCENTER

このチェック タイプでは、vCenter 設定を監査できます。

```
<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>
```

次は、合格した vSphere 監査例です。



キーワード

次の表に、VMware コンプライアンス チェックにおける各キーワードの使用方法を示します。

キーワード	使用例とサポートされている設定
type	このキーワードは、監査ファイル内の特定項目によって実行されるチェック タイプを示します。VMware 監査は、次の 3 つの監査チェック タイプを実行できます。 • AUDIT_VM • AUDIT_ESX • AUDIT_VCENTER
description	このキーワードは、実行するチェックを簡単に説明するのに使用します。**description** フィールドは一意である必要があり、2 つのチェックが同じ **description** フィールドを持つことはできません。これは、SecurityCenter が **description** フィールドを使用して、プラグイン ID 番号を自動生成するためです。 例: `description: "Disconnect unauthorized devices - 'floppyX.present = false'"`

info	このキーワードは、実行するチェックに関する詳細な説明を追加するのに使用します。指定できる **info** フィールド数に制限はありません。info コンテンツは、二重引用符で囲む必要があります。 例: `info: "Make sure floppy drive is not attached"`
regex	このキーワードは、特定の正規表現と一致する項目を検索します。 例: `regex: "floppy([Xx]|[0-9]+)\\.present :"`

チェックのコンプラインスは、**expect** キーワードまたは **not_expect** キーワードのいずれかと、チェック出力との比較により判断されます。1 つのチェック内で複数のコンプライアンス テスト タグを使用することはできません。

キーワード	使用例とサポートされている設定
expect	このキーワードにより、**regex** キーワードによって一致する構成項目を監査できます。また、**regex** キーワードが使用されていない場合、構成全体から **expect** 文字列が検索されます。 regex **によって検出された構成行が** expect **文字列と一致している場合、チェックは合格になります。**または **regex** が設定されていない場合は、**expect** 文字列が構成から検出された場合のみ合格になります。 例: `regex: "floppy([Xx]|[0-9]+)\\.present :"` `expect: floppy([Xx]|[0-9]+)\\.present : false"` または: `expect: floppy([Xx]|[0-9]+)\\.present : false"` 上記の例では、**expect** キーワードは、フロッピー ドライブが存在していないことをチェックしています。
not_expect	このキーワードにより、構成内に含まれてはならない構成項目を検索できます。 これは **expect** の反対の動作になります。**regex** によって検出された構成行が **not_expect** 文字列と一致しない場合、チェックは合格になります。また、**regex** キーワードが設定されていない場合は、構成に **not_expect** 文字列が検出されなければ合格です。 例: `regex: floppy([Xx]|[0-9]+)\\.present : "` `not_expect: floppy([Xx]|[0-9]+)\\.present : false"`

	または: `not_expect: floppy([Xx]|[0-9]+)\\.present : false"` 上記の例では、**expec** キーワードは、フロッピー ドライブが存在していないことをチェックしています。

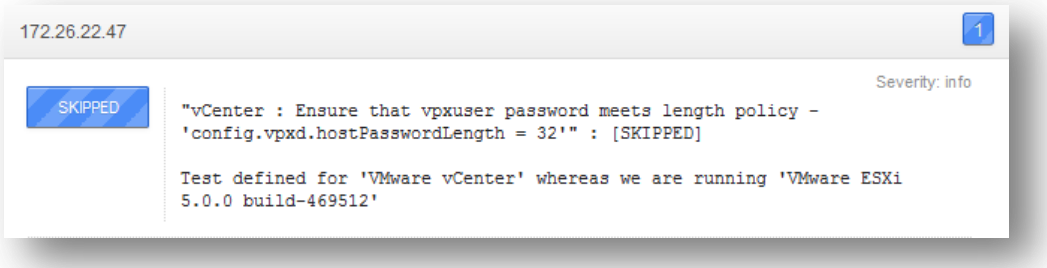
追加メモ

チェックに合格すると、このプラグインは、ポリシーに一致するすべての VM をレポートします。Tenable によって提供される監査は、VM 名とターゲットの IP の両方をレポートします。ただし、VM の IP アドレスは、VMware ツールがインストールされていなければレポートされません。

次に、レポートの例を示します。

```
Test VM 2, poweredOff (toolsNotInstalled) - vmsafe.enable : NOT found
Test VM Audit (172.26.23.123) - vmsafe.enable : NOT found
```

ESX/ESXi と vCenter の両方を同じポリシーでスキャンできます。ただし、ESX/ESXi ホストに対して実行される vCenter チェックはスキップされます。



詳細情報

Tenable では、Nessus のインストール、展開、構成、ユーザー操作、総合的なテストについて詳述したドキュメントを多数用意しています。

- **Nessus 5.2 インストールおよび構成ガイド** – インストールおよび構成の詳しい手順
- **Nessus 5.2 ユーザー ガイド** – Nessus ユーザー インタフェースの構成および操作方法
- **Unix と Windows での Nessus 認証チェック** – Nessus 脆弱性スキャナを使用して認証ネットワーク スキャンを実行する方法
- **Nessus コンプライアンス チェック** – Nessus と SecurityCenter を使用するコンプライアンス チェックについて理解して実行するためのハイレベル ガイド
- **Nessus v2 File Format** – Nessus 3.2 と NessusClient 3.2 で導入された **.nessus** ファイル フォーマットの構造について説明しています。
- **Nessus 5.0 REST Protocol Specification** – Nessus の REST プロトコルとインタフェースについて説明しています。

- **Nessus 5 and Antivirus** – いくつかの有名なセキュリティ ソフトウェア パッケージが Nessus とどう連動するかについて概要を説明し、セキュリティの侵害や脆弱性スキャンの妨害を招くことなく、より良い共存を可能にするためのヒントや解決方法を示します。
- **Nessus 5 and Mobile Device Scanning** – Nessus が Microsoft Active Directory およびモバイル デバイス管理サーバーと統合されて、ネットワーク上で使用されているモバイル デバイスを特定する方法について説明しています。
- **Nessus 5.0 and Scanning Virtual Machines** – Tenable Network Security の Nessus 脆弱性スキャナを使用して仮想プラットフォームの構成およびそこで実行されているソフトウェアを監査する方法について説明しています。
- **Strategic Anti-malware Monitoring with Nessus, PVS, and LCE** – Tenable の USM プラットフォームが悪意のあるさまざまなソフトウェアを検知し、マルウェアを特定し、その感染範囲を判定する方法について説明しています。
- **Patch Management Integration** – Nessus と SecurityCenter が、IBM TEM、Microsoft WSUS と SCCM、VMware Go、および Red Hat Network Satellite パッチ管理システム上の資格情報を活用して、Nessus スキャナで資格情報が利用できない可能性があるシステム上でパッチ監査を実行する方法について説明しています。
- **Real-Time Compliance Monitoring** – さまざまなタイプの政府規制や財政規制に適合するために、Tenable のソリューションがどのように使用できるかを説明しています。
- **Tenable Products Plugin Families** – Nessus、Log Correlation Engine、Passive Vulnerability Scanner 用のプラグイン ファミリーの説明とサマリーが記されています。
- **SecurityCenter Administration Guide**

その他のオンライン リソースは以下のとおりです。

- Nessus ディスカッション フォーラム：https://discussions.nessus.org/
- Tenable ブログ：http://www.tenable.com/blog
- Tenable ポッドキャスト：http://www.tenable.com/podcast
- 使用例のビデオ：http://www.youtube.com/user/tenablesecurity
- Tenable Twitter ページ：http://twitter.com/tenablesecurity

support@tenable.com または sales@tenable.com までお気軽にお問い合わせください。または Tenable のウェブ サイト http://www.tenable.com/ をご参照ください。

付録 A：Unix コンプライアンス ファイルの例

注：次のファイル **tenable_unix_compliance_template.audit** は、Tenable サポート ポータル (https://support.tenable.com/) からダウンロードできます。このファイルには、Tenable の Unix コンプライアンス モジュールを使用して実行できるさまざまなタイプの Unix コンプライアンス チェックが含まれています。実際のファイルには、ここには反映されていない更新が含まれる可能性があります。

```
#
# (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>
```

付録 B：Windows コンプライアンス ファイルの例

注：次のファイルは、Tenable サポート ポータル (https://support.tenable.com/) からダウンロードできます。実際のファイルには、ここには反映されていない更新が含まれる可能性があります。このスクリプト名は **financial_microsoft_windows_user_audit_guideline_v2.audit** で、ユーザー管理用の一般的な強化ガイドに基づいています。このポリシーは、合理的なパスワード ポリシー、アカウント ロックアウト ポリシーを検索し、ログイン イベントが Windows イベント ログに記録されていることを確認します。

```
# (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>
```

付録 C：XSLT による .audit 変換

コンプライアンス チェック プラグインの中には、Palo Alto、VMware、および Unix コンプライアンス チェックなど、XML コンテンツの監査に依存するものがあります。これらの機能を正しく利用するには、XSLT の作成に慣れていることが有益です。場合によっては、XSLT の構築は、何度か試してみることが必要です。このプロセスに慣れたところで、次のステップとして **.audit** への変換を行いますが、これは直感的な動作ではありません。この付録では、カスタム XSLT の構築と利用、および **.audit** ファイルへの変換について正しい方法を解説します。

複数の監査チェック (AUDIT_XML、AUDIT_VCENTER、AUDIT_ESX など) は個別の存在ですが、同じ基盤ロジックを使用します。XML の基本操作を理解することにより、これらのチェックを、XML を利用する他のプラットフォームに直接変換できるようになります。

ユーザーは **xsltproc** ユーティリティを使用して 次の 7 つのステップに従うことにより、XML コンテンツ用のカスタム **.audit** ファイルを生成できます。

ステップ 1：xsltproc をインストールする

xsltproc がシステムにインストールされているかどうか確認し、インストールされていない場合は、インストールします。インストールと動作状態を確認するには、次のコマンドを入力します。

```
[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
[..]
```

ステップ 2：使用する XML ファイルを識別する

使用する XML ファイルを決定します。ファイルの場所と、これが XML コンテンツであることを確認します。たとえば、次のとおりです。

```
[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>
[..]
```

ステップ 3：XSLT と XPath について理解する

このプロセスには、XSLT と XPath の基本的な理解が必要です。それぞれ詳細は以下を参照してください。

- w3schools.com - XSLT – Transformation
- w3schools.com - XPath Introduction

ステップ 4：XSLT を作成する

次のステップでは、XSLT を使用して XML ファイルから関連データを抽出します。まずは XSLT を作成します。これは、ファイルから関連データを抽出するのに必要です。たとえば、XML から "name" 要素を抽出する必要があるとします。次の XSLT により、必要な情報を抽出します。

```
<?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>
```

XSLT を作成したら、次のステップでテストを行うのに適した場所に保存します。このサンプル ファイルは **pa.xsl** として保存します。

.audit 内のカスタム XSLT を使用する場合、最初の 3 行と最後の 2 行は無視します。標準的なこれらの行は Nessus プラグイン **nbin** によって追加されます。この例では、5 ～ 8 行が AUDIT_XML 項目または AUDIT_REPORTS 項目で使用される行になります。

前提または新しいテクニックを検証する XSLT を構築する際、ステップ 5 のテスト プロセスも使用できます。XSLT に詳しくなかったり、複雑な変換を行う場合、このプロセスは特に有益です。

ステップ 5：XSLT の動作を検証する

xsltproc で XSLT の動作を検証します。一般的なフォーマットは次のとおりです。

```
/usr/bin/xsltproc {XSLT file} {Source XML}
```

上記のステップでサンプル ファイル名にプラグインすると、次のデータが返されます。このデータにより、XSLT が正しく、適切なフォーマットであり、期待データが返されていることを確認できます。

```
[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
```

ステップ 6：XSLT を .audit にコピーする

XSLT の動作に問題がないことを確認できたら、目的の XSLT 行 (この例の 5 ～ 8 行) を **.audit** チェックにコピーします。

```
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>"
```

カスタム XSLT の各行は、二重引用符で囲まれた独自の **xsl_stmt** 要素内に配置する必要があります。**xslt_stmt** 要素が二重引用符を使用して <xsl> 文をカプセル化するので、その他使用する二重引用符は、エスケープする必要があります。**二重引用符のエスケープは重要です。このことを行っていないと、チェック実行中、エラーが発生する危険性があります。**

```
/usr/bin/xsltproc {XSLT file} {Source XML}
```

次のステップで、適切にエスケープされた二重引用符の例をいくつか紹介します。

ステップ 7：監査準備完了

これまでの 6 つのステップが完了したら、監査を構築するために必要なすべてが終了となります。

```
<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>
```

Tenable Network Security について

Tenable Network Security は、新たに生じる脆弱性、脅威、コンプライアンス関係のリスクに事前に対応するために、米国防総省全体、および多数の世界最大級の企業と政府をはじめとする 24,000 以上の組織に信頼され、利用されています。同社の Nessus および SecurityCenter ソリューションは、脆弱性の特定、攻撃の防止、および多数の規制要件へのコンプライアンスに対する標準を設定し続けます。詳細については、www.tenable.com をご参照ください。

グローバル本社

Tenable Network Security
7021 Columbia Gateway Drive
Suite 500
Columbia, MD 21046
410.872.0555
www.tenable.com