Nessus-Referenzhandbuch für Compliancetests

10. Januar 2014

(Revision 39)

Inhaltsverzeichnis

Einleitung

Das vorliegende Dokument beschreibt die Syntax zur Erstellung angepasster **`.audit`**-Dateien, unter deren Verwendung Sie Audits für UNIX-, Windows-, Datenbank-, SCADA-, IBM iSeries- und Cisco-Systeme nach einer Compliancerichtlinie durchführen und verschiedene Systeme nach sensiblen Inhalten durchsuchen können.

Dieses Handbuch soll Sie bei der manuellen Erstellung von Compliance-Auditdateien und beim Verständnis der Syntax unterstützen. Eine grundsätzliche Beschreibung der Funktionsweise von Tenable-Compliancetests entnehmen Sie dem Handbuch „Nessus Compliance Checks“ („Nessus-Compliancetests“), das im PDF-Format vom Tenable Support Portal heruntergeladen werden kann.

Zwar sind Audits für SCADA-Systeme mit Nessus möglich, doch ist diese Funktionalität nicht Gegenstand des vorliegenden Dokuments. Weitere Informationen hierzu entnehmen Sie der Tenable SCADA-Informationsseite (hier klicken).

Voraussetzungen

In diesem Dokument werden bestimmte Kenntnisse zum Nessus-Sicherheitslückenscanner sowie ein umfassendes Verständnis der zu prüfenden Zielsysteme vorausgesetzt. Weitere Informationen zur Konfiguration von Nessus für lokale Patchaudits unter UNIX und Windows entnehmen Sie dem Handbuch „Nessus Credentials Checks for Unix and Windows“ („Authentifizierte Nessus-Tests für UNIX und Windows“), das unter http://www.tenable.com/products/nessus/documentation verfügbar ist.

Regeln und Konventionen

In der gesamten Dokumentation werden Dateinamen, Daemons und ausführbare Dateien in einer Schriftart wie **`courier bold`** angezeigt.

Befehlszeichenoptionen und Schlüsselwörter werden ebenfalls in der Schriftart **`courier bold`** angezeigt. Die Befehlszeilen sind teils mit, teils ohne Befehlszeilen-Prompt und den Ausgabetext des betreffenden Befehls aufgeführt. In den Befehlszeilen erscheint der ausgeführte Befehl in der Schriftart **`courier bold`**, um zu verdeutlichen, was der Benutzer eingegeben hat. Die vom System generierte Beispielausgabe ist hingegen in der Schriftart `courier` (ohne Fettdruck) aufgeführt. Es folgt ein Beispiel für die Ausführung des UNIX-Befehls **`pwd`**:

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

Wichtige Hinweise und Aspekte werden durch dieses Symbol und graue Textfelder hervorgehoben.

Tipps, Beispiele und Best Practices (Empfehlungen) werden durch dieses Symbol und weißen Text auf blauem Grund hervorgehoben.

Referenz zu Compliancedateien für Windows-Konfigurationsaudits

Den Windows **`.audit`**-Compliancedateien liegt eine speziell formatierte Textdatei zugrunde. Einträge in dieser Datei können eine Vielzahl angepasster Objekttests (z. B. Tests von Registrierungseinstellungen) wie auch allgemeinere Tests – etwa von Einstellungen in der lokalen Sicherheitsrichtlinie – aufrufen. Zur Verdeutlichung werden in diesem Handbuch durchgängig Beispiele verwendet.

Verwendung von Anführungszeichen:

Um Auditfelder gesetzte einzelne bzw. doppelte Anführungszeichen werden austauschbar verwendet. Ausgenommen sind lediglich die folgenden Fälle:

1. Bei Windows-Compliancetests, in denen Sonderfelder (z. B. CRLF) literal interpretiert werden müssen, müssen einzelne Anführungszeichen gesetzt werden. Eingebettete Felder, die als Strings zu interpretieren sind, müssen mit Escapezeichen versehen werden.

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

2. Doppelte Anführungszeichen sind erforderlich, wenn „include_paths“ und „exclude_paths“ für WindowsFiles verwendet werden.

Wenn Strings in einem beliebigen Feldtyp `description`, `value_data`, `regex` usw.) verwendet werden, die doppelte Anführungszeichen aufweisen, kann auf zweierlei Weise verfahren werden:

a. Verwenden Sie den jeweils anderen Anführungszeichentyp als äußere umschließende Anführungszeichen.

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

b. Stellen Sie eingebetteten doppelten Anführungszeichen den umgekehrten Schrägstrich („\“) als Escapezeichen voran.

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

Testtyp

Alle Windows-Compliancetests müssen in die Kapselung **`check_type`** gesetzt werden, die mit der Typangabe „Windows“ und der Versionsangabe „2“ zu versehen ist:

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

Einen exemplarischen Windows-Compliancetest finden Sie in Anhang B. Er beginnt mit der **`check_type`**-Einstellung „Windows“ und der Versionsangabe „2“ und endet mit dem Tag „**`</check_type>`**“.

Dies ist erforderlich, um **`.audit`**-Dateien für Windows von jenen für UNIX (oder anderen Plattformen) zu unterscheiden.

Wertdaten

Die Syntax von **`.audit`**-Dateien enthält Schlüsselwörter, denen zur Testanpassung verschiedene Werttypen zugeordnet werden können. Dieser Abschnitt beschreibt diese Schlüsselwörter und das Format der einzugebenden Daten.

Datentypen

Folgende Datentypen können bei Tests eingegeben werden:

Datentyp	Beschreibung
DWORD	0 bis 2.147.483.647
RANGE [X..Y]	Hierbei ist X ein DWORD-Wert oder MIN und Y ein DWORD-Wert oder MAX.

Beispiele:

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

Außerdem können Zahlen mit den Vorzeichen „+“ oder „–“ versehen und als Hexadezimalwerte ausgewiesen werden. Auch Hexadezimalwerte können mit Vorzeichen versehen werden. Nachfolgend sind gültige Beispiele in einem REGISTRY_SETTING-Audit für ein POLICY_DWORD angegeben (die in Klammern stehenden Angaben dienen nur der Veranschaulichung):

```
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)
```

Komplexe Ausdrücke

Komplexe Ausdrücke verwenden Sie wie folgt für das **`value_data`**-Feld:

- ||: logisches ODER
- &&: logisches UND
- |: binäres ODER (auf Bitebene)
- &: binäres UND (auf Bitebene)
- (und): Trennzeichen in komplexen Ausdrücken

Beispiele:

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

Das Feld „check_type“

Dieses Feld unterscheidet sich von dem zuvor beschriebenen Feld „**`check_type`**“, das am Anfang jeder Auditdatei steht, um den generischen Audittyp („Windows“, „WindowsFiles“, „Unix“, „Database“, „Cisco“) zu bezeichnen. Es ist optional und kann auf der Basis von **`value_data`**-Werten für Windows verwendet werden, um den auszuführenden Test zu bestimmen. Folgende Einstellungen sind möglich:

- CHECK_EQUAL: Vergleicht den Remotewert mit dem Richtlinienwert (Standard, sofern **`check_type`** nicht angegeben ist).
- CHECK_EQUAL_ANY: Überprüft, ob jedes Element in **`value_data`** mindestens einmal in der Systemliste vorhanden ist.
- CHECK_NOT_EQUAL: Überprüft, ob sich der Remotewert vom Richtlinienwert unterscheidet.
- CHECK_GREATER_THAN: Überprüft, ob der Remotewert größer als der Richtlinienwert ist.

- CHECK_GREATER_THAN_OR_EQUAL: Überprüft, ob der Remotewert größer als oder gleich dem Richtlinienwert ist.
- CHECK_LESS_THAN: Überprüft, ob der Remotewert kleiner als der Richtlinienwert ist.
- CHECK_LESS_THAN_OR_EQUAL: Überprüft, ob der Remotewert kleiner als oder gleich dem Richtlinienwert ist.
- CHECK_REGEX: Überprüft, ob der Remotewert dem regulären Ausdruck im Richtlinienwert entspricht (funktioniert nur bei POLICY_TEXT und POLICY_MULTI_TEXT).
- CHECK_SUBSET: Überprüft, ob die Remote-ACL eine Teilmenge der Richtlinien-ACL ist (funktioniert nur mit ACLs).
- CHECK_SUPERSET: Überprüft, ob die Remote-ACL eine Obermenge der Richtlinien-ACL ist (funktioniert nur mit ACLs, die Berechtigungen ablehnen).

Nachfolgend gezeigt ist ein Beispielaudit, mit dem überprüft wird, ob der Kontoname „Guest“ („Gast“) für ein Gästekonto nicht vorhanden ist.

```
<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>
```

Ist ein Wert ungleich „Guest“ vorhanden, dann ist der Test erfolgreich; wird hingegen „Guest“ gefunden, dann schlägt er fehl.

Das Feld „group_policy“

Das Feld „**group_policy**“ kann zur Angabe eines kurzen Textstrings verwendet werden, der das Audit beschreibt. **group_policy** muss in der Auditdatei enthalten und nach dem Feld **check_type** eingefügt sein.

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

…

</group_policy>
</check_type>
```

Das Feld „info“

Mit dem optionalen Feld „**info**“ kann jedes Auditfeld mit einem oder mehreren externen Verweisen versehen werden. Beispielsweise wird dieses Feld zum Einfügen von Referenzen aus NIST CCE-Tags sowie CIS-spezifischen Auditanforderungen verwendet. Diese externen Referenzen werden im von Nessus ausgeführten Audit ausgegeben und erscheinen im Nessus-Bericht bzw. auf der Benutzeroberfläche von SecurityCenter.

Das nächste Beispiel zeigt eine Kennwortauditrichtlinie, die so angepasst wurde, dass sie Verweise auf eine (fiktive) Unternehmensrichtlinie enthält:

```
<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>
```

Wenn mehrere Verweise auf Richtlinien im selben Audit erforderlich sind, kann im vom Schlüsselwort „**info**“ angegebenen String das Trennzeichen „\n“ zur Angabe mehrerer Strings benutzt werden. Sehen Sie sich hierzu folgendes Audit an:

```
<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>
```

Die Ausführung dieses Audits mit dem Befehlszeilentool **nasl** führt zur folgenden Ausgabe:

```
# /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"
```

Das Feld „debug“

Mithilfe des optionalen Feldes „**debug**“ können Probleme bei Compliancetests von Windows-Inhalten analysiert werden. Das Schlüsselwort „debug“ gibt Informationen zum durchgeführten Inhaltsscan aus. Diese umfassen etwa die verarbeiteten und gescannten Dateien und ggf. aufgetretene Ergebnisse. Aufgrund der umfangreichen Ausgabe bei Verwendung dieses Schlüsselworts sollte es nur zur Fehlersuche verwendet werden. Beispiel:

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

ACL-Format

In diesem Abschnitt wird die Syntax beschrieben, mit der festgestellt werden kann, ob eine Datei oder ein Ordner die gewünschte ACL-Einstellung aufweist.

Tests für Dateizugriffssteuerungen

Syntax

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

Eine Zugriffssteuerungsliste (Access Control List, ACL) für Dateien ist mit dem Schlüsselwort **`file_acl`** gekennzeichnet. Zur Verwendung mit einem Dateiberechtigungselement muss die ACL einen eindeutigen Namen aufweisen. Eine Datei-ACL kann einen oder mehrere Benutzereinträge enthalten.

Zugeordnete Typen	Zulässige Typen
`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`**	Diese Einstellungen sind optional. Generische Rechte sind: • full control • modify • read & execute • read • write • list folder contents Erweiterte Rechte sind: • 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

Das folgende Beispiel zeigt den **`.audit`**-Text einer Dateizugriffssteuerung:

```
 <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>
```

Tests für Registrierungszugriffssteuerungen

Syntax

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

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

</acl>
```

Eine Registrierungs-ACL ist mit dem Schlüsselwort **registry_acl** gekennzeichnet. Zur Verwendung mit einem Registrierungsberechtigungselement muss die ACL einen eindeutigen Namen aufweisen. Eine Registrierungs-ACL kann einen oder mehrere Benutzereinträge enthalten.

Zugeordnete Typen	Zulässige Typen
acl_inheritance	• not inherited • inherited • not used
acl_apply	• this key only • this key and subkeys • subkeys only
acl_allow **acl_deny**	Diese Einstellungen sind optional und dienen der Definition von Rechten, die ein Benutzer für das Objekt hat. Generische Rechte sind: • full control • read Erweiterte Rechte sind: • full control • query value • set value • create subkey • enumerate subkeys • notify • create link • delete • write dac • write owner • read control

Das folgende Beispiel zeigt eine Registrierungs-ACL in einer **.audit**-Datei:

```
<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>
```

Tests für Dienstzugriffssteuerungen

Syntax

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

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

</acl>
```

Eine Dienst-ACL ist mit dem Schlüsselwort **`service_acl`** gekennzeichnet. Zur Verwendung mit einem Dienstberechtigungselement muss die ACL einen eindeutigen Namen aufweisen. Eine Dienst-ACL kann einen oder mehrere Benutzereinträge enthalten.

Zugeordnete Typen	Zulässige Typen
`acl_inheritance`	• not inherited • inherited • not used
`acl_apply`	• this object only
`acl_allow` **`acl_deny`**	Diese Einstellungen sind optional und dienen der Definition von Rechten, die ein Benutzer für das Objekt hat. Generische Rechte sind: • full control • read • start, stop and pause • write • delete Erweiterte Rechte sind:

	• 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

Das folgende Beispiel zeigt einen Dienst-ACL-Test:

```
 <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>
```

Tests für Launch-Berechtigungssteuerungen

Syntax

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

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

</acl>
```

Eine Launch-ACL (Start-ACL) ist mit dem Schlüsselwort **`launch_acl`** gekennzeichnet. Zur Verwendung mit einem DCOM-Launch-Berechtigungselement muss die ACL einen eindeutigen Namen aufweisen. Eine Launch-ACL kann einen oder mehrere Benutzereinträge enthalten.

Zugeordnete Typen	Zulässige Typen
acl_inheritance	• not inherited • inherited
acl_apply	• this object only
acl_allow **acl_deny**	Diese Einstellungen sind optional und dienen der Definition von Rechten, die ein Benutzer für das Objekt hat. Generische Rechte sind: • local launch • remote launch • local activation • remote activation

Diese ACL kann nur auf Windows XP-, 2003- und Vista-Systemen (sowie teilweise auf Windows 2000-Systemen) verwendet werden.

Das folgende Beispiel zeigt einen Launch-ACL-Test:

```
<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>
```

Tests für Launch2-Berechtigungssteuerungen

Syntax

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

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

</acl>
```

Eine Launch2-ACL ist mit dem Schlüsselwort **launch2_acl** gekennzeichnet. Zur Verwendung mit einem DCOM-Launch-Berechtigungselement muss die ACL einen eindeutigen Namen aufweisen. Eine Launch2-ACL kann einen oder mehrere Benutzereinträge enthalten.

Zugeordnete Typen	Zulässige Typen
acl_inheritance	• not inherited • inherited
acl_apply	• this object only
acl_allow **acl_deny**	Diese Einstellungen sind optional und dienen der Definition von Rechten, die ein Benutzer für das Objekt hat. Generische Rechte sind: • launch

Die Launch2-ACL kann nur auf Windows 2000- und Windows NT-Systemen verwendet werden.

Das folgende Beispiel zeigt einen Launch-ACL-Test:

```
 <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>
```

Tests für Zugriffsberechtigungssteuerungen

Syntax

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

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

</acl>
```

Eine Zugriffs-ACL ist mit dem Schlüsselwort **`access_acl`** gekennzeichnet. Zur Verwendung mit einem DCOM-Zugriffsberechtigungselement muss die ACL einen eindeutigen Namen aufweisen. Eine Zugriffs-ACL kann einen oder mehrere Benutzereinträge enthalten.

Zugeordnete Typen	Zulässige Typen
`acl_inheritance`	• not inherited • inherited
`acl_apply`	• this object only
`acl_allow` **`acl_deny`**	Diese Einstellungen sind optional und dienen der Definition von Rechten, die ein Benutzer für das Objekt hat. Generische Rechte sind: • local access • remote access

Das folgende Beispiel zeigt einen Zugriffs-ACL-Test:

```
 <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>
```

Benutzerdefinierte Elemente

Ein benutzerdefiniertes Element ist ein vollständiger Test, der auf der Basis der oben definierten Schlüsselwörter definiert wird. Die folgende Liste enthält die verfügbaren Typen für benutzerdefinierte Elemente. Jeder Test beginnt mit dem Tag „**<custom_item>**“ und endet mit „**</custom_item>**“. In die Tags eingeschlossen ist eine Liste mit einem oder mehreren Schlüsselwörtern, die vom Compliancetest-Parser zur Durchführung des Tests interpretiert werden.

Bei den benutzerdefinierten Audittests können „**</custom_item>**“ und „**</item>**“ als schließende Tags beliebig gegeneinander ausgetauscht werden.

PASSWORD_POLICY

Syntax

```
<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>
```

Dieses Richtlinienelement prüft auf die unter „Windows-Einstellungen“ > „Sicherheitseinstellungen“ > „Kontorichtlinien“ > „Kennwortrichtlinie“ definierten Werte.

Der Test wird nach Aufruf der Funktion **NetUserModalsGet** mit der Stufe 1 ausgeführt.

Diese Elemente benutzen das Feld **password_policy**, um zu beschreiben, welches Element der Kennwortrichtlinie geprüft werden muss. Zulässige Typen sind:

- ENFORCE_PASSWORD_HISTORY („Kennwortchronik erzwingen“)
  ```
  value_type: POLICY_DWORD
  value_data: DWORD oder RANGE [Anzahl gespeicherter Kennwörter]
  ```
- MAXIMUM_PASSWORD_AGE („Maximales Kennwortalter“)
  ```
  value_type: TIME_DAY
  value_data: DWORD oder RANGE [Zeit in Tagen]
  ```
- MINIMUM_PASSWORD_AGE („Minimales Kennwortalter“)
  ```
  value_type: TIME_DAY
  value_data: DWORD oder RANGE [Zeit in Tagen]
  ```
- MINIMUM_PASSWORD_LENGTH („Minimale Kennwortlänge“)
  ```
  value_type: POLICY_DWORD
  value_data: DWORD oder RANGE [Mindestanzahl der Zeichen im Kennwort]
  ```

- COMPLEXITY_REQUIREMENTS („Kennwort muss Komplexitätsvoraussetzungen entsprechen“)
  `value_type: POLICY_SET`
  `value_data: "Enabled" oder "Disabled"`

- REVERSIBLE_ENCRYPTION („Kennwörter mit umkehrbarer Verschlüsselung für alle Benutzer in der Domäne speichern“)
  `value_type: POLICY_SET`
  `value_data: "Enabled" oder "Disabled"`

- FORCE_LOGOFF („Netzwerksicherheit: Abmeldung nach Ablauf der Anmeldezeit erzwingen“)
  `value_type: POLICY_SET`
  `value_data: "Enabled" oder "Disabled"`

Es gibt gegenwärtig keinen Test für die Richtlinie „Kennwort mit umkehrbarer Verschlüsselung für alle Benutzer in der Domäne speichern“.

Die Richtlinie FORCE_LOGOFF finden Sie unter „Sicherheitseinstellungen“ > „Lokale Richtlinien“ > „Sicherheitsoptionen“.

Das folgende Beispiel zeigt ein Kennwortrichtlinienaudit:

```
<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

Syntax

```
<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>
```

Dieses Richtlinienelement prüft auf die unter „Sicherheitseinstellungen“ > „Kontorichtlinien“ > „Kontosperrungsrichtlinien“ definierten Werte.

Der Test wird nach Aufruf der Funktion **`NetUserModalsGet`** mit der Stufe 3 ausgeführt.

Dieses Element benutzt das Feld **`lockout_policy`**, um zu beschreiben, welches Element der Kennwortrichtlinie geprüft werden muss. Zulässige Typen sind:

- LOCKOUT_DURATION („Kontosperrdauer“)
  `value_type: TIME_MINUTE`
  `value_data: DWORD oder RANGE [Zeit in Minuten]`

- LOCKOUT_THRESHOLD („Kontosperrschwellwert“)
  `value_type: POLICY_DWORD`
  `value_data: DWORD oder RANGE [Zeit in Tagen]`

- LOCKOUT_RESET („Zurücksetzen des Kontosperrzählers nach“)
  `value_type: TIME_MINUTE`
  `value_data: DWORD oder RANGE [Zeit in Minuten]`

Beispiel:

```
<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

Syntax

```
<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>
```

Dieses Richtlinienelement prüft auf die unter „Sicherheitseinstellungen“ > „Kontorichtlinien“ > „Kerberos-Richtlinie“ definierten Werte.

Der Test wird nach Aufruf der Funktion **`NetUserModalsGet`** mit der Stufe 1 ausgeführt.

Dieses Element benutzt das Feld **`kerberos_policy`**, um zu beschreiben, welches Element der Kennwortrichtlinie geprüft werden muss. Zulässige Typen sind:

- USER_LOGON_RESTRICTIONS („Benutzeranmeldeeinschränkungen erzwingen“)
  `value_type: POLICY_SET`
  `value_data: "Enabled" oder "Disabled"`

- SERVICE_TICKET_LIFETIME („Max. Gültigkeitsdauer des Diensttickets“)
  `value_type: TIME_MINUTE`
  `value_data: DWORD oder RANGE [Zeit in Minuten]`

- USER_TICKET_LIFETIME („Max. Gültigkeitsdauer des Benutzertickets“)
  `value_type: TIME_HOUR`
  `value_data: DWORD oder RANGE [Zeit in Stunden]`

- USER_TICKET_RENEWAL_LIFETIME („Max. Zeitraum, in dem ein Benutzerticket erneuert werden kann“)
  `value_type: TIME_DAY`
  `value_data: DWORD oder RANGE [Zeit in Tagen]`

- CLOCK_SYNCHRONIZATION_TOLERANCE („Max. Toleranz für die Synchronisation des Computertakts“)
  `value_type: TIME_MINUTE`
  `value_data: DWORD oder RANGE [Zeit in Minuten]`

Die Kerberos-Richtlinie kann erst nach Authentifizierung bei einem KDC (Key Distribution Center, Schlüsselverteilcenter) getestet werden. Unter Windows agiert in der Regel ein Domänencontroller als KDC.

Beispiel:

```
<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

Syntax

```
<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>
```

Dieses Richtlinienelement prüft auf die unter „Sicherheitseinstellungen“ > „Lokale Richtlinien“ > „Überwachungsrichtlinie“ definierten Werte.

Der Test wird nach Aufruf der Funktion **`LsaQueryInformationPolicy`** mit der Stufe **`PolicyAuditEventsInformation`** ausgeführt.

Dieses Element benutzt das Feld **`audit_policy`**, um zu beschreiben, welches Element der Kennwortrichtlinie geprüft werden muss. Zulässige Typen sind:

- AUDIT_ACCOUNT_LOGON („Anmeldeversuche prüfen“)
- AUDIT_ACCOUNT_MANAGER („Kontoverwaltung prüfen“)
- AUDIT_DIRECTORY_SERVICE_ACCESS („Verzeichnisdienstzugriff prüfen“)
- AUDIT_LOGON („Anmeldeereignisse prüfen“)
- AUDIT_OBJECT_ACCESS („Objektzugriffsversuche prüfen“)
- AUDIT_POLICY_CHANGE („Richtlinienänderungen prüfen“)
- AUDIT_PRIVILEGE_USE („Berechtigungseinsatz prüfen“)
- AUDIT_DETAILED_TRACKING („Prozessverfolgung prüfen“)

- AUDIT_SYSTEM („Systemereignisse prüfen“)

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

Beachten Sie, dass das Leerzeichen in „Success, Failure“ erforderlich ist.

Beispiel:

```
<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

Syntax

```
<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>
```

Dieses Richtlinienelement prüft auf die in auditpol /get /category:* aufgeführten Werte.

Dieser Test wird durch Ausführen von „cmd.exe auditpol /get /category:*“ über die WMI durchgeführt.

Dieses Element benutzt das Feld **`audit_policy_subcategory`**, um festzustellen, welche Unterkategorie geprüft werden muss. Als SUBCATEGORY_POLICY_TYPE zulässig sind:

- Security State Change („Sicherheitsstatusänderung“)
- Security System Extension („Sicherheitssystemerweiterung“)
- System Integrity („Systemintegrität“)
- IPsec Driver („IPsec-Treiber“)
- Other System Events („Andere Systemereignisse“)
- Logon („Anmeldung“)
- Logoff („Abmeldung“)
- Account Lockout („Kontosperre“)
- IPsec Main Mode („IPsec-Hauptmodus“)
- IPsec Quick Mode („IPsec-Schnellmodus“)
- IPsec Extended Mode („IPsec-Erweiterungsmodus“)
- Special Logon („Spezielle Anmeldung“)
- Other Logon/Logoff Events („Andere Anmelde-/Abmeldeereignisse“)
- Network Policy Server („Netzwerkrichtlinienserver“)
- File System („Dateisystem“)

- Registry („Registrierung“)
- Kernel Object („Kernelobjekt“)
- SAM
- Certification Services („Zertifizierungsdienste“)
- Application Generated („Anwendung wurde generiert“)
- Handle Manipulation („Handleänderung“)
- File Share („Dateifreigabe“)
- Filtering Platform Packet Drop („Filterplattform: Verworfene Pakete“)
- Filtering Platform Connection („Filterplattformverbindung“)
- Other Object Access Events („Andere Objektzugriffsereignisse“)
- Sensitive Privilege Use („Verwendung sensibler Rechte“)
- Non Sensitive Privilege Use („Verwendung nicht sensibler Rechte“)
- Other Privilege Use Events („Andere Rechteverwendungsereignisse“)
- Process Creation („Prozesserstellung“)
- Process Termination („Prozessbeendigung“)
- DPAPI Activity („DPAPI-Aktivität“)
- RPC Events („RPC-Ereignisse“)
- Audit Policy Change („Richtlinienänderungen prüfen“)
- Authentication Policy Change („Authentifizierungsrichtlinienänderung“)
- Authorization Policy Change („Autorisierungsrichtlinienänderung“)
- MPSSVC Rule-Level Policy Change („MPSSVC-Richtlinienänderung auf Regelebene“)
- Filtering Platform Policy Change („Filterplattform-Richtlinienänderung“)
- Other Policy Change Events („Andere Richtlinienänderungsereignisse“)
- User Account Management („Benutzerkontoverwaltung“)
- Computer Account Management („Computerkontoverwaltung“)
- Security Group Management („Sicherheitsgruppenverwaltung“)
- Distribution Group Management („Verteilergruppenverwaltung“)
- Application Group Management („Anwendungsgruppenverwaltung“)
- Other Account Management Events („Andere Kontoverwaltungsereignisse“)
- Directory Service Access („Verzeichnisdienstzugriff“)
- Directory Service Changes („Verzeichnisdienständerungen“)
- Directory Service Replication („Verzeichnisdienstreplikation“)
- Detailed Directory Service Replication („Detaillierte Verzeichnisdienstreplikation“)
- Credential Validation („Überprüfung der Anmeldeinformationen“)
- Kerberos Service Ticket Operations („Ticketvorgänge des Kerberos-Diensts“)
- Other Account Logon Events („Andere Kontoanmeldungsereignisse “)

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

Beachten Sie, dass das Leerzeichen in „Success, Failure“ erforderlich ist.

Dieser Test kann nur bei Systemen unter Windows Vista/2008 Server und höher verwendet werden. Wenn eine Firewall aktiviert ist, müssen Sie nicht nur die WMI als Ausnahme in den Firewalleinstellungen hinzufügen, sondern auch über **`gpedit.msc`** den Eintrag „Windows-Firewall: Eingehende Remoteverwaltungsausnahme zulassen“ aktivieren. Dieser Test funktioniert möglicherweise nicht auf nicht englischsprachigen Systemen unter Windows Vista/2008 Server, auf denen „auditpol“ nicht installiert ist.

Beispiel:

```
<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

Syntax

```
<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:\Programme\Microsoft\Exchange
Server\ExShell.psc1"
</custom_item>
```

Bei diesem Test wird **powershell.exe** auf dem Remoteserver mit den Argumenten ausgeführt, die mit „**powershell_args**“ übergeben wurden. Die Befehlsausgabe wird zurückgegeben, wenn „**only_show_cmd_output**“ den Wert „YES“ hat, oder mit „**value_data**“ verglichen, sofern „**value_data**“ angegeben wurde.

Zugeordnete Typen:

Dieses Element verwendet das Feld „**powershell_args**“ zur Angabe der Argumente, die an **powershell.exe** übergeben werden müssen. Wenn **powershell.exe** nicht am Standardspeicherort vorhanden ist, müssen Sie zur Angabe des Speicherorts das Schlüsselwort **powershell_console_file** verwenden. Zurzeit werden nur „`get-`“ Cmdlets unterstützt. Beispiel:

- `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`

Dieses Element verwendet das optionale Feld „**only_show_cmd_output**“, wenn die gesamte Befehlsausgabe gemeldet werden muss:

- `only_show_cmd_output: YES` oder `NO`

Zusätzliche Anmerkungen:

1. Wenn Sie „**only_show_cmd_output**“ festlegen und den Schweregrad der Ausgabe bezeichnen möchten, können Sie diesen Schweregrad mit dem Tag „severity“ ändern. Standardwert ist „INFO“.
2. PowerShell ist unter einigen Windows-Betriebssystemen (z. B. Windows XP oder Windows 2003) nicht standardmäßig installiert. Deswegen führt dieser Test auf solchen Systemen zu keinen Ergebnissen. Sie müssen deswegen vor Verwendung dieses Tests sicherstellen, dass PowerShell auf dem Zielremotesystem installiert ist.
3. Damit dieser Test korrekt funktioniert, muss der WMI-Dienst aktiviert sein. Konfigurieren Sie bei der Firewall außerdem die Option „Eingehende Remoteverwaltungsausnahme zulassen“.
4. Cmdlet-Aliase (z. B. „`gps`“ statt „`Get-Process`“) sind unzulässig.

Beispiel:

Im folgenden Beispiel wird das PowerShell-Cmdlet „Get-Hotfix“ ausgeführt, es wird ein where-Objekt angegeben, damit keine Hotfixes mit der ID „File 1“ ausgewählt werden, und abschließend werden die Parameter „Description“, „HotfixID“ und „Installedby“ als Liste formatiert ausgegeben.

```
<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>
```

Beispiel:

Im nächsten Beispiel wird überprüft, ob der Windows-Dienst „WinRM“ ausgeführt wird.

```
<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

Syntax

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

Dieser Test führt **powershell.exe** auf dem Remoteserver zusammen mit den angegebenen Informationen zum Vergleich des erwarteten Datei-Hash mit dem Hash der Datei im System aus.

Zusätzliche Anmerkungen:

- Es wird standardmäßig ein MD5-Hash der Datei verglichen. Benutzer können jedoch mit den Algorithmen SHA1, SHA256, SHA384, SHA512 oder RIPEMD160 generierte Hashes vergleichen.
- Für diesen Test muss PowerShell installiert und WMI auf dem Zielsystem aktiviert sein.

Beispiel:

Das folgende Beispiel vergleicht den angegebenen MD5-Hash mit dem Datei-Hash von `C:\test\test2.zip`.

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

Beispiel:

Das folgende Beispiel vergleicht den angegebenen SHA1-Hash mit dem Datei-Hash von `C:\test\test3.zip`.

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

AUDIT_IIS_APPCMD

Syntax

```
<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 oder NO
 (optional) check_type: [CHECK_TYPE]
 (optional) severity: ["HIGH" oder "MEDIUM" oder "LOW"]
</custom_item>
```

Bei diesem Test wird **appcmd.exe** auf einem Server ausgeführt, auf dem IIS läuft. Die Argumente werden mit „**appcmd_args**“ angegeben, und die Compliance wird durch den Vergleich der Ausgabe mit **value_data** bestimmt. In einigen Fällen (z. B. bei Listenkonfigurationen) kann es wünschenswert sein, nur die Befehlsausgabe in den Bericht zu integrieren. In solchen Fällen muss „**only_show_cmd_output**“ verwendet werden.

Dieser Test steht nur für Version 7 der Internetinformationsdienste (Internet Information Services, IIS) unter Windows zur Verfügung.

Dieses Element verwendet das Feld „**appcmd_args**“ zur Angabe der Argumente, die an **appcmd.exe** übergeben werden müssen. Derzeit können nur „list“-Befehle angegeben werden.

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

Dieses Element verwendet das optionale Feld „**only_show_cmd_output**“, wenn die gesamte Befehlsausgabe in den Bericht eingeschlossen werden muss.

Beispiel:

Bei diesem Test wird das Ergebnis von „`appcmd.exe list AppPools /processModel.identityType:ApplicationPoolIdentity`“ mit **value_data** verglichen. Der Test ist nur erfolgreich, wenn `'APPPOOL "DefaultAppPool"'` in der Ausgabe enthalten ist.

```
<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

Syntax

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

Dieser Test fragt die Liste der offenen TCP/UDP-Ports auf dem Zielsystem ab und vergleicht sie mit einer Liste zulässiger Ports. Der Test ruft anhand der Ausgaben von „netstat -ano“ oder „netstat -an“ eine Liste offener Ports ab und überprüft dann, ob diese Ports tatsächlich offen sind, indem er den Portstatus mithilfe von (get_port_state()/get_udp_port_state()) verifiziert.

Anmerkungen:

- **value_data** akzeptiert auch einen regulären Ausdruck als Portbereichsangabe. Insofern werden Ausdrücke wie „8[0-9]+“ erfolgreich verarbeitet.

Beispiele:

Das folgende Beispiel vergleicht „**value_data**“ mit einer Liste der auf dem Ziel offenen TCP-Ports:

```
<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>
```

Das folgende Beispiel vergleicht „**value_data**“ mit einer Liste der auf dem Ziel offenen UDP-Ports:

```
<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

Syntax

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

Dieser Test fragt die Liste der offenen TCP/UDP-Ports auf dem Zielsystem ab und vergleicht sie mit einer Liste der abgelehnten Ports. Der Test ruft anhand der Ausgaben von „netstat -ano“ oder „netstat -an“ eine Liste offener Ports ab und überprüft dann, ob diese Ports tatsächlich offen sind, indem er den Portstatus mithilfe von (get_port_state()/get_udp_port_state()) verifiziert.

Zulässige Typen sind:

- `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`

Anmerkungen:

- **value_data** akzeptiert auch einen regulären Ausdruck als Portbereichsangabe. Insofern werden Ausdrücke wie „8[0-9]+“ erfolgreich verarbeitet.

Beispiele:

Das folgende Beispiel vergleicht „**value_data**“ mit einer Liste der auf dem Ziel offenen TCP-Ports.

```
<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>
```

Das folgende Beispiel vergleicht „**value_data**“ mit einer Liste der auf dem Ziel offenen UDP-Ports.

```
<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

Syntax

```
<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>
```

Dieser Test fragt über den jeweiligen Port ausgeführten Prozess ab. Der Test bestimmt anhand der Ausgabe von „netstat -ano“ und „tasklist /svc“, welcher Prozess über welchen TCP/UDP-Port ausgeführt wird.

Zulässige Typen sind:

- `value_type: POLICY_TEXT`
- `value_data:` Beliebiger String, z. B. „`foo.exe`“.
- `port_type: TCP` oder `UDP`
- `port_no:` Portnummer, z. B. 80, 445
- `port_option: CAN_BE_CLOSED`

Anmerkungen:

- Wenn **`port_option`** auf CAN_BE_CLOSED festgelegt ist, gibt der Test das Ergebnis PASS („Bestanden“) zurück, wenn der Port auf dem Remotesystem nicht offen ist; andernfalls wird ein Fehler ausgegeben.

- Windows 2000 und ältere Systeme unterstützen „netstat -ano“ nicht. Daher funktioniert der Test erst ab Windows XP.

Beispiele:

Das folgende Beispiel prüft, ob der über TCP-Port 5900 ausgeführte Prozess entweder „vss.exe“ oder „vssrvc.exe“ ist.

```
<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>
```

Folgendes Beispiel ähnelt dem ersten Beispiel. Allerdings wird in diesem Beispiel „check_type“ verwendet.

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

CHECK_ACCOUNT

Syntax

```
<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>
```

Dieses Richtlinienelement prüft die unter „Sicherheitseinstellungen“ > „Lokale Richtlinien“ > „Sicherheitsoptionen“ definierten Werte.

- Accounts: Administrator account status („Konten: Administratorkontostatus“)
- Accounts: Guest account status („Konten: Gastkontostatus“)
- Accounts: Rename administrator account („Konten: Administratorkonto umbenennen“)
- Accounts: Rename guest account („Konten: Gastkonto umbenennen“)

Dieser Test wird durch einen Aufruf der Funktion **`LsaQueryInformationPolicy`** ausgeführt. Für den Abruf der Domänen-/System-SID wird die Stufe **`PolicyAccountDomainInformation`**, für den Abruf der Administrator- und Gastnamen die Stufe **`LsaLookupSid`** und für den Abruf der Kontoinformationen die Stufe **`NetUserGetInfo`** angegeben.

Dieses Element benutzt das Feld **`account_type`**, um zu beschreiben, welches Konto geprüft werden muss. Zulässige Typen sind:

- ADMINISTRATOR_ACCOUNT („Konten: Administratorkontostatus“)
  `value_type: POLICY_SET`
  `value_data: "Enabled" oder "Disabled"`
- GUEST_ACCOUNT („Konten: Gastkontostatus“)
  `value_type: POLICY_SET`
  `value_data: "Enabled" oder "Disabled"`
- ADMINISTRATOR_ACCOUNT („Konten: Administratorkonto umbenennen“)
  `value_type: POLICY_TEXT`
  `value_data: "TEXT HERE" [Administratorname]`
  `check_type: [CHECK_TYPE] (einer der möglichen check_type-Werte)`
- GUEST_ACCOUNT („Konten: Gastkonto umbenennen“)
  `value_type: POLICY_TEXT`
  `value_data: "TEXT HERE" [Gastname]`
  `check_type: [CHECK_TYPE] (einer der möglichen check_type-Werte)`

Abhängig von dem für die Domänen vorgesehenen Teil der Anmeldedaten werden die lokalen Systemkonten oder die Domänenkonten geprüft.

Beispiele:

```
<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

Syntax

```
<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>
```

Dieses Richtlinienelement testet Gruppennamen und Status der in **`lusmgr.msc`** aufgeführten Gruppen.

Dieses Element benutzt das Feld **`group_type`**, um zu beschreiben, welches Konto geprüft werden muss. Zulässige Typen sind:

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

Zulässige Typen für das Feld **`value_type`** sind:

- POLICY_SET (Status der Gruppe wird geprüft)
  `value_type: POLICY_SET`
  `value_data: "Enabled" oder "Disabled"`
- POLICY_TEXT (Name der Gruppe wird geprüft)
  `value_type: POLICY_TEXT`
  `value_data: "Guests1"` (In diesem Fall kann **`value_data`** ein beliebiger Textstring sein.)

Beispiele:

```
<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

Syntax

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

Dieses Richtlinienelement prüft den unter „Sicherheitseinstellungen“ > „Lokale Richtlinien“ > „Sicherheitsoptionen“ > „Netzwerkzugriff: Anonyme SID-/Namensübersetzung zulassen“ definierten Wert. Der Test wird nach Aufruf der Funktion **`LsaQuerySecurityObject`** für den LSA-Richtlinienhandle ausgeführt.

Zulässige Typen sind:

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

Wenn Sie dieses Audit verwenden, beachten Sie bitte, dass diese Richtlinie:

- eine Berechtigungsprüfung für den LSA-Dienst ist,
- überprüft, ob bei ANONYMOUS_USER das Flag POLICY_LOOKUP_NAMES gesetzt ist,
- in Windows 2003 veraltet ist, weil ein anonymer Benutzer nicht auf die LSA-Pipe zugreifen kann.

Beispiel:

```
 <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

Für diesen Test muss der Remote-Registrierungszugriff auf dem Windows-Remotesystem einwandfrei funktionieren.

Syntax

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

Dieses Richtlinienelement prüft die unter „System Services“ („Systemdienste“) definierten Startwerte. Der Test wird nach Aufruf der Funktion **`RegQueryValueEx`** für die folgenden Schlüssel ausgeführt:

- Schlüssel: "SYSTEM\CurrentControlSet\Services\" + service_name
- Element: "Start"

Zulässige Typen sind:

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

Das Feld **`service_name`** ist dabei der echte Dienstname. Dieser kann wie folgt abgerufen werden:

1. Starten Sie das Systemsteuerungs-Applet „Dienste“ (unter „Verwaltung“).
2. Wählen Sie den gewünschten Dienst aus.
3. Öffnen Sie das Dialogfeld „Eigenschaften“ (Rechtsklick auf den Dienst, dann „Eigenschaften“ auswählen).
4. Der echte Dienstname ist unter „Dienstname“ angegeben.

Die Dienstberechtigungseinstellung kann mit einem SERVICE_PERMISSIONS-Element geprüft werden.

Beispiel:

```
 <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

Syntax

```
<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>
```

Mit diesem Richtlinienelement wird geprüft, ob eine bestimmte Liste von Benutzern in mindestens einer Gruppe vorhanden ist.

Der zulässige Typ ist:

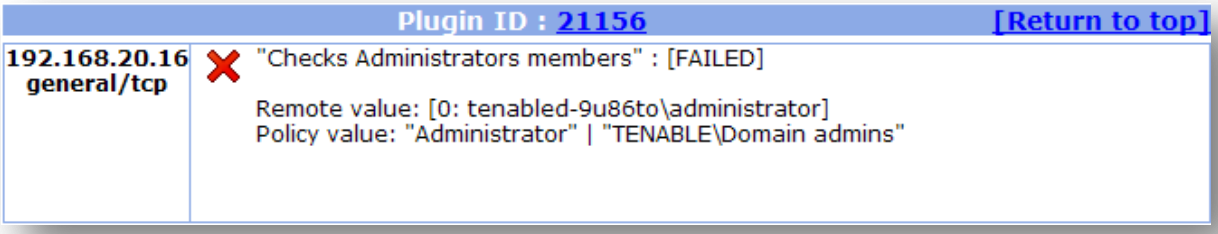
```
value_type: POLICY_TEXT or POLICY_MULTI_TEXT
value_data: "user1" && "user2" && ... && "usern"
```

Bei der Verwendung dieses Audits müssen Sie beachten, dass ein Benutzername mit dem Domänennamen angegeben werden kann (z. B. „MYDOMAIN\John Smith“) und das Feld **group_name** eine einzelne Gruppe bezeichnet, die geprüft werden soll.

Mit derselben Nessus-**.audit**-Datei können mehrere verschiedene Kundenelemente überprüft werden, was Audits von Benutzerlisten in mehreren Gruppen sehr einfach macht. Das nächste Beispiel zeigt eine **.audit**-Richtlinie, die überprüft, ob alle Mitglieder der Gruppe „Administratoren“ Benutzer mit den Werten „Administrator“ und „TENABLE\Domain admins“ sind:

```
 <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>
```

Die folgende Bildschirmabbildung zeigt exemplarisch die Ausführung des obigen **.audit**-Dateiinhalts für einen Windows 2003-Server:



USER_GROUPS_POLICY

Syntax

```
<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>
```

Mit diesem Richtlinienelement wird geprüft, ob ein Windows-Benutzer zu den in **value_data** angegebenen Gruppen gehört. Bei Verwendung dieses Audits können Sie nur Domänenbenutzer eines Domänencontrollers testen. Für integrierte Benutzer wie „Local Service“ kann dieser Test nicht verwendet werden.

Beispiel:

```
 <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

Syntax

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

Dieses Richtlinienelement prüft den folgenden in „Sicherheitseinstellungen“ > „Lokale Richtlinien“ > „Zuweisen von Benutzerrechten“ definierten Wert. Der Test wird nach Aufruf der Funktion **`LsaEnumerateAccountsWithUserRight`** für den LSA-Richtlinienhandle ausgeführt.

Das Feld **`right_type`** nennt dabei die zu testende Berechtigung. Zulässige Werte sind:

```
right_type: RIGHT
```

Hierbei kann **`RIGHT`** folgende Werte annehmen:

```
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
```

Der zulässige Typ ist:

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

Mit Benutzerrechtstests können viele Anforderungen an Domänencontroller überprüft werden. Diese Tests müssen in eine separate Richtliniendatei eingeschlossen sein und dürfen nur für den Domänencontroller und genau ein System in der Domäne ausgeführt werden.

Der Typ **`right`** darf nicht in Anführungszeichen gesetzt werden, da er als Token verarbeitet wird.

Beispiel:

```
 <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

Für diesen Test muss der Remote-Registrierungszugriff auf dem Windows-Remotesystem einwandfrei funktionieren.

Syntax

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

Mit dieser Richtlinie wird geprüft, ob die Datei (**`value_data`**) vorhanden ist oder nicht (**`file_option`**). Der Test wird nach dem Aufruf der Funktion **`CreateFile`** ausgeführt.

Zulässige Typen sind:

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

Beispiele:

```
 <custom_item>
  type: FILE_CHECK
  description: "Check that win.ini exists in the system root"
  value_type: POLICY_TEXT
```

```
  value_data: "%SystemRoot%\win.ini"
  file_option: MUST_EXIST
 </custom_item>
```

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

FILE_VERSION

Für diesen Test muss der Remote-Registrierungszugriff auf dem Windows-Remotesystem einwandfrei funktionieren.

Syntax

```
<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>
```

Mit diesem Richtlinienelement wird standardmäßig getestet, ob die Version der im Feld **`file`** angegebenen Datei größer als oder gleich der Remotedateiversion ist. Der Test kann zudem verwendet werden, um durch Angabe der Option **`check_type`** festzustellen, ob die Dateiversion kleiner ist.

Zulässige Typen sind:

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

Beispiele:

```
 <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

Für diesen Test muss der Remote-Registrierungszugriff auf dem Windows-Remotesystem einwandfrei funktionieren.

Syntax

```
<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>
```

Mit diesem Richtlinienelement wird überprüft, ob die ACL „FILE_PERMISSIONS“ korrekt ist. Der Test wird nach Aufruf der Funktion **`GetSecurityInfo`** mit der Stufe 7 für den Dateihandle ausgeführt.

Der zulässige Typ ist:

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

Die folgenden vordefinierten Pfade können im Datei- bzw. Ordnernamen verwendet werden:

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

Wenn Sie dieses Audit verwenden, beachten Sie bitte Folgendes:

- Das Feld **`file`** muss den vollständigen Pfad zur Datei oder zum Ordner enthalten (z. B. **`C:\WINDOWS\SYSTEM32`**) oder die oben aufgeführten Schlüsselwörter verwenden. Bei der Verwendung von Pfadschlüsselwörtern muss die Remote-Registrierung aktiviert sein, damit Nessus die Pfadvariablenwerte bestimmen kann.
- Das Feld **`value_data`** enthält den Namen einer ACL, die in der Richtliniendatei definiert ist.

- Das Feld **acl_option** kann auf „CAN_BE_NULL“ gesetzt werden, damit der Test auch erfolgreich ist, wenn die Datei nicht vorhanden ist; mit dem Wert „CAN_NOT_BE_NULL“ hingegen wird die umgekehrte Logik konfiguriert.

Beispiele:

```
<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>
```

Wenn der obige Test ausgeführt wird, überprüft das Compliancemodul, ob die für „**%SystemRoot%\SYSTEM32**“ definierten Berechtigungen mit den in „file_acl ACL1“ beschriebenen übereinstimmen.

FILE_AUDIT

Für diesen Test muss der Remote-Registrierungszugriff auf dem Windows-Remotesystem einwandfrei funktionieren.

Syntax

```
<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>
```

Mit diesem Richtlinienelement werden die unter „Eigenschaften“ > „Sicherheit“ > „Erweitert“ > „Überwachung“ festgelegten Auditeigenschaften einer Datei oder eines Ordners unter Verwendung der angegebenen ACL überprüft. Der Test wird nach Aufruf der Funktion **`GetSecurityInfo`** mit der Stufe „SACL_SECURITY_INFORMATION“ für den Dateihandle ausgeführt.

Der zulässige Typ ist:

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

Die folgenden vordefinierten Pfade können im Datei- bzw. Ordnernamen verwendet werden:

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

Wenn Sie dieses Audit verwenden, beachten Sie bitte Folgendes:

- Das Feld **`file`** muss den vollständigen Pfad zur Datei oder zum Ordner enthalten (z. B. **`C:\WINDOWS\SYSTEM32`**) oder die oben aufgeführten Schlüsselwörter verwenden. Bei der Verwendung von Pfadschlüsselwörtern muss die Remote-Registrierung aktiviert sein, damit Nessus die Pfadvariablenwerte bestimmen kann.
- Das Feld **`value_data`** enthält den Namen der ACL, die in der Richtliniendatei definiert ist.
- Das Feld **`acl_option`** kann auf „CAN_BE_NULL“ gesetzt werden, damit der Test auch erfolgreich ist, wenn die Datei nicht vorhanden ist; mit dem Wert „CAN_NOT_BE_NULL“ hingegen wird die umgekehrte Logik konfiguriert.
- Die Felder **`acl_allow`** und **`acl_deny`** entsprechen den Auditereignissen „Successful“ („Erfolgreich“) bzw. „Failed“ („Fehlgeschlagen“).

Das folgende Beispiel zeigt eine **`.audit`**-Datei, die die Funktion „FILE_AUDIT“ einschließlich einer ACL-Regel namens „ACL1“ implementiert.

```
<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

Für diesen Test muss der Remote-Registrierungszugriff auf dem Windows-Remotesystem einwandfrei funktionieren.

Syntax

```
<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>
```

Mit diesem Richtlinienelement wird geprüft, ob die Datei den regulären Ausdruck **`regex`** enthält und dieser Ausdruck dem Feld **`expect`** entspricht.

Der Test wird nach Aufruf der Funktion **`ReadFile`** für den Dateihandle ausgeführt.

Die Datei wird über SMB in einen Pufferspeicher auf dem Nessus-Server eingelesen. Dieser Puffer wird dann auf Compliance geprüft. Die Dateien werden nicht auf der Festplatte des Nessus-Servers gespeichert, sondern lediglich zur Analyse in den Puffer kopiert.

Der zulässige Typ ist:

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

Die folgenden vordefinierten Pfade können im Datei- bzw. Ordnernamen verwendet werden:

```
%allusersprofile%
%windir%
%systemroot%
%commonfiles%
```

```
%programfiles%
%systemdrive%
```

Wenn Sie diesen Audittyp verwenden, beachten Sie bitte Folgendes:

- Das Feld **value_data** muss den vollständigen Pfad zur Datei oder zum Ordner enthalten (z. B. **C:\WINDOWS\SYSTEM32**) oder die oben aufgeführten Schlüsselwörter verwenden. Bei der Verwendung von Pfadschlüsselwörtern muss die Remote-Registrierung aktiviert sein, damit Nessus die Pfadvariablenwerte bestimmen kann.
- Mit dem Feld **regex** wird überprüft, ob ein Element in der Datei vorhanden ist.
- Mit dem Feld **expect** wird überprüft, ob das Element dem regulären Ausdruck entspricht.
- Das Feld **file_option** kann auf „CAN_BE_NULL“ gesetzt werden, damit der Test auch erfolgreich ist, wenn die Datei nicht vorhanden ist.
- Das Feld **file_option** kann auf „CAN_NOT_BE_NULL“ gesetzt werden, damit der Test auch dann fehlschlägt, wenn die Datei vorhanden, aber leer ist.
- Mithilfe des Feldes **avoid_floppy_access** kann festgelegt werden, dass beim Audit kein Test durchgeführt wird, in dessen Verlauf ein Zugriff auf das Diskettenlaufwerk erfolgen würde. Verwenden Sie diese Option, wenn im Rahmen eines Audits ein Zugriff auf ein Diskettenlaufwerk erfolgen würde, in dem keine Diskette liegt.

Beispiel:

```
 <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

Für diesen Test muss der Remote-Registrierungszugriff auf dem Windows-Remotesystem einwandfrei funktionieren.

Syntax

```
<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>
```

Mit diesem Richtlinienelement wird geprüft, ob die Datei den regulären Ausdruck „regex“ enthält und dieser Ausdruck nicht mit dem Feld „expect“ übereinstimmt. Der Test wird nach Aufruf der Funktion **ReadFile** für den Dateihandle ausgeführt.

Der zulässige Typ ist:

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

Die folgenden vordefinierten Pfade können im Datei- bzw. Ordnernamen verwendet werden:

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

Wenn Sie diesen Audittyp verwenden, beachten Sie bitte Folgendes:

- Das Feld **value_data** muss den vollständigen Pfad zur Datei oder zum Ordner enthalten (z. B. **C:\WINDOWS\SYSTEM32**) oder die oben aufgeführten Schlüsselwörter verwenden. Bei der Verwendung von Pfadschlüsselwörtern muss die Remote-Registrierung aktiviert sein, damit Nessus die Pfadvariablenwerte bestimmen kann.
- Mit dem Feld **regex** wird überprüft, ob ein Element in der Datei vorhanden ist.
- Mit dem Feld **expect** wird überprüft, ob das Element dem regulären Ausdruck entspricht.
- Das Feld **file_option** kann auf „CAN_BE_NULL“ gesetzt werden, damit der Test auch erfolgreich ist, wenn die Datei nicht vorhanden ist.
- Das Feld **file_option** kann auf „CAN_NOT_BE_NULL“ gesetzt werden, damit der Test auch dann fehlschlägt, wenn die Datei vorhanden, aber leer ist.

Beispiel:

```
<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

Für diesen Test muss der Remote-Registrierungszugriff auf dem Windows-Remotesystem einwandfrei funktionieren.

Syntax

```
<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>
```

Mit dieser Richtlinie wird geprüft, ob der Registrierungsschlüssel (oder das Element) vorhanden ist oder nicht. Der Test wird nach dem Aufruf der Funktionen **RegOpenKeyEx** und **RegQueryValueEx** ausgeführt.

Zulässige Typen sind:

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

Wenn das Feld **key_item** nicht angegeben wird, prüft dieses Element auf Vorhandensein des Registrierungspfades, andernfalls auf Vorhandensein des angegebenen Elements.

Beispiel:

```
 <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

Für diesen Test muss der Remote-Registrierungszugriff auf dem Windows-Remotesystem einwandfrei funktionieren.

Syntax

```
<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>
```

Mit diesem Richtlinienelement wird der Wert eines Registrierungsschlüssels geprüft. Das Richtlinienelement wird von vielen Richtlinientests unter „Sicherheitseinstellungen“ > „Lokale Richtlinien“ > „Sicherheitsoptionen“ verwendet. Der Test wird nach dem Aufruf der Funktion **`RegQueryValueEx`** ausgeführt.

Das Feld **`reg_key`** enthält den Namen des Registrierungsschlüssels (z. B. „HKLM\SOFTWARE\Microsoft\Driver Signing“). Der erste Teil des Schlüssels („HKLM“) dient der Verbindung mit der richtigen Registrierungsstruktur, der nachfolgende Pfad ist eine statische Angabe der Position von **`reg_item`**.

Die Struktur „HKU“ („HKEY_USERS“) ist ein Spezialfall: Die Angabe einer SID für HKU-Schlüssel ist nicht möglich. Aus diesem Grund iteriert die `nbin`-Datei intern über jede SID und ist nur dann erfolgreich, wenn der Wert in jeder SID gültig ist.

Beispiel:

```
<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>
```

würde iterieren über:

```
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
...
```

Der Test ist erfolgreich, wenn das Element „ScreenSaveActive“ für alle SIDs auf „1“ steht.

Das optionale Feld **`reg_option`** kann auf „CAN_BE_NULL“ gesetzt werden, damit der Test auch erfolgreich ist, wenn der Schlüssel nicht vorhanden ist; mit dem Wert „CAN_NOT_BE_NULL“ hingegen wird die umgekehrte Logik konfiguriert.

Die zusätzliche Option **`reg_enum`** mit dem Argument „ENUM_SUBKEYS“ kann verwendet werden, um einen angegebenen Wert für alle Unterschlüssel eines Registrierungsschlüssels aufzulisten. Im Schlüssel **`HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall`** beispielsweise sind viele Softwarepakete aufgelistet. Wenn Sie den „CurrentVersion“-Wert für alle Unterschlüssel von „Uninstall“ vergleichen möchten, verwenden Sie **`reg_enum`**.

Beispiel:

```
<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>
```

Dieses Audit der HKU-Registrierungsstruktur schließt die SID (Sicherheits-ID) nicht in den Registrierungspfad **reg_key** ein. Im folgenden Beispiel wird nach allen HUK-SIDs für den angegebenen **reg_item**-Wert gesucht.

Beispiel:

```
<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>
```

Die folgenden Hauptfeldtypen für **value_type** sind vorhanden:

- POLICY_SET
  value_data: "Enabled" oder "Disabled"
- POLICY_DWORD
  value_data: DWORD oder RANGE [gleiches DWORD wie in Registrierung oder Bereich]
- POLICY_TEXT
  value_data: "TEXT" [gleicher Text wie in der Registrierung]
- POLICY_MULTI_TEXT
  value_data: "TEXT1" && "TEXT2" && ... && "TEXTN" [gleicher Wortlaut wie in der Registrierung]
- POLICY_BINARY
  value_data: "0102ac0b...34fb" [gleiche Binärdatei wie in der Registrierung]
- FILE_ACL, REG_ACL, SERVICE_ACL, LAUNCH_ACL, ACCESS_ACL
  value_data: "acl_name" [Name der zu verwendenden ACL]

Die folgenden optionalen Feldtypen für **value_type** sind vorhanden und werden in vordefinierten Elementen verwendet:

- 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" oder "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" oder "send ntlmv2 response only\refuse lm & ntlm"`
- LDAPCLIENT_SET
  `value_data: "None", "Negotiate Signing" oder "Require Signing"`
- EVENT_METHOD
  `value_data: "by days", "manually" oder "as needed"`
- POLICY_DAY
  `value_data: DWORD oder RANGE [Zeit in Tagen]`
- POLICY_KBYTE
  `value_data: DWORD oder RANGE`

Verwenden Sie für das Feld **`custom_item`** den **`value_type`**-Hauptwert. Optionale Typen wurden für vordefinierte Elemente erstellt.

Wenn **`value_type`** eine ACL ist, muss das Registrierungselement eine Sicherheitsbeschreibung im Binärformat sein.

Beispiele:

```
<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

Für diesen Test muss der Remote-Registrierungszugriff auf dem Windows-Remotesystem einwandfrei funktionieren.

Syntax

```
<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>
```

Mit diesem Richtlinienelement wird überprüft, ob die ACL für Registrierungsschlüssel korrekt ist. Der Test wird nach Aufruf der Funktion **`RegGetKeySecurity`** für den Registrierungsschlüsselhandle ausgeführt.

Der zulässige Typ ist:

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

Die folgenden vordefinierten Pfade können im Feld **`reg_key`** verwendet werden:

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

Wenn Sie dieses Audit verwenden, beachten Sie bitte Folgendes:

- Das Feld **`reg_key`** muss den vollständigen Pfad zum Dateiregistrierungsschlüssel enthalten.
- Das Feld **`value_data`** enthält den Namen einer ACL, die in der Richtliniendatei definiert ist.
- Das Feld **`acl_option`** kann auf „CAN_BE_NULL“ gesetzt werden, damit der Test auch erfolgreich ist, wenn der Schlüssel nicht vorhanden ist; mit dem Wert „CAN_NOT_BE_NULL“ hingegen wird die umgekehrte Logik konfiguriert.

Beispiel:

```
<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>
```

Wenn der obige Test ausgeführt wird, überprüft das Compliancemodul, ob die für „**`HKLM\SOFTWARE\Microsoft`**“ definierten Berechtigungen mit den in „registry_acl ACL2“ beschriebenen übereinstimmen.

REGISTRY_AUDIT

Für diesen Test muss der Remote-Registrierungszugriff auf dem Windows-Remotesystem einwandfrei funktionieren.

Syntax

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

Mit diesem Richtlinienelement wird überprüft, ob die ACL für Registrierungsschlüssel korrekt ist. Der Test wird nach Aufruf der Funktion **`RegGetKeySecurity`** für den Registrierungsschlüsselhandle ausgeführt.

Der zulässige Typ ist:

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

Der folgende vordefinierte Pfad kann im Feld **`reg_key`** verwendet werden:

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

Wenn Sie dieses Audit verwenden, beachten Sie bitte Folgendes:

- Das Feld **`reg_key`** muss den vollständigen Pfad zum Dateiregistrierungsschlüssel enthalten.
- Das Feld **`value_data`** enthält den Namen der ACL, die in der Richtliniendatei definiert ist.
- Das Feld **`acl_option`** kann auf „CAN_BE_NULL“ gesetzt werden, damit der Test auch erfolgreich ist, wenn der Schlüssel nicht vorhanden ist; mit dem Wert „CAN_NOT_BE_NULL“ hingegen wird die umgekehrte Logik konfiguriert.
- Die Felder **`acl_allow`** und **`acl_deny`** entsprechen den Auditereignissen „Successful“ bzw. „Failed“.

Das nächste Beispiel zeigt eine **`.audit`**-Datei, die den Registrierungsschlüssel „HKLM\SOFTWARE\Microsoft“ mit einer hier nicht angezeigten ACL namens „ACL2“ vergleicht:

```
<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

Für diesen Test muss der Remote-Registrierungszugriff auf dem Windows-Remotesystem einwandfrei funktionieren.

Syntax

```
<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>
```

Mit diesem Richtlinienelement wird der Wert eines Registrierungsschlüsseltyps geprüft. Der Test wird nach dem Aufruf der Funktion **`RegQueryValue`** ausgeführt.

Das Feld **`reg_key`** enthält den Namen des Registrierungsschlüssels (z. B. „HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon“). Der erste Teil des Schlüssels (HKLM, HKU, HKCU usw.) dient der Verbindung mit der richtigen Registrierungsstruktur. In den meisten Fällen erfordert das Feld **`reg_key`** einen statischen Registrierungseintrag ohne Platzhalter, aber beim Suchen nach Werten in HKU (HKEY_USERS) sind Ausnahmen zulässig. Wenn in HKU ein Pfad angegeben wird, durchläuft der Suchvorgang alle Benutzerwerte in HKU auf der Suche nach dem Wert im angegebenen Pfad. Wenn z. B. `reg_key: "HKU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"` zusammen mit dem **`reg_item`** „brastk“ angegeben wird, werden alle Benutzer in HKU nach dem Registrierungsschlüsselwert „brastk“ im jeweiligen Pfad durchsucht: „HKU\<user_id>\SOFTWARE\Microsoft\Windows\CurrentVersion\Run“. Beispiel:

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

Dieser Test sucht in:

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

Das optionale Feld **`reg_option`** kann auf „CAN_BE_NULL“ festgelegt werden, damit der Test auch erfolgreich ist, wenn der Schlüssel nicht vorhanden ist; mit dem Wert „CAN_NOT_BE_NULL“ hingegen wird die umgekehrte Logik konfiguriert.

Für diesen Test steht nur POLICY_TEXT **`value_type`** zur Verfügung.

Es folgt ein Beispiel für eine **`.audit`**-Datei, die den Registrierungstyp von „HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon“ prüft:

```
<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>
```

Beachten Sie, dass Audits von HKCU auf vielen Installationen von Windows möglicherweise nicht funktionieren. Sie benötigen dazu die Schlüssel der aktuellen Benutzer („ Current User“), die in der Regel nicht vorhanden sind, wenn Nessus sich über SMB authentifiziert. Als Lösung sind Audits von HKU (d. h. aller Benutzer) möglich. Wenn das Plugin erkennt, dass ein HKU-Schlüssel geprüft wird, durchsucht es automatisch alle verfügbaren SIDs mit Ausnahme des Schlüssels `.DEFAULT`. Der Nachteil dieser Vorgehensweise besteht darin, dass es auch alle Systembenutzer prüft (z. B. SYSTEM, NT Authority, usw.). Verwenden Sie zum Ausschließen dieser Benutzer `reg_ignore_hku_users`. Beispiel:

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

Dies funktioniert nur mit dem Test `REGISTRY_SETTING`.

SERVICE_PERMISSIONS

Syntax

```
<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>
```

Mit diesem Richtlinienelement wird überprüft, ob die Dienst-ACL korrekt ist. Der Test wird nach Aufruf der Funktion **`QueryServiceObjectSecurity`** für den Diensthandle ausgeführt.

Der zulässige Typ ist:

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

Wenn Sie dieses Audit verwenden, beachten Sie bitte Folgendes:

- Das Feld **`value_data`** enthält den Namen einer ACL, die in der Richtliniendatei definiert ist.
- Das Feld **`acl_option`** kann auf „CAN_BE_NULL“ gesetzt werden, damit der Test auch erfolgreich ist, wenn der Schlüssel nicht vorhanden ist; mit dem Wert „CAN_NOT_BE_NULL“ hingegen wird die umgekehrte Logik konfiguriert.

Beispiel:

```
 <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>
```

Wenn der obige Test ausgeführt wird, überprüft das Compliancemodul, ob die für den Warndienst definierten Berechtigungen mit den in „ACL3“ beschriebenen übereinstimmen.

SERVICE_AUDIT

Syntax

```
<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>
```

Mit diesem Richtlinienelement wird überprüft, ob die Dienst-ACL korrekt ist. Der Test wird nach Aufruf der Funktion **`QueryServiceObjectSecurity`** für den Diensthandle ausgeführt.

Der zulässige Typ ist:

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

Wenn Sie diesen Audittyp verwenden, beachten Sie bitte Folgendes:

- Das Feld **`value_data`** enthält den Namen der ACL, die in der Richtliniendatei definiert ist.
- Das Feld **`acl_option`** kann auf „CAN_BE_NULL“ gesetzt werden, damit der Test auch erfolgreich ist, wenn der Schlüssel nicht vorhanden ist; mit dem Wert „CAN_NOT_BE_NULL“ hingegen wird die umgekehrte Logik konfiguriert.
- Die Felder **`acl_allow`** und **`acl_deny`** entsprechen den Auditereignissen „Successful“ bzw. „Failed“.

Das folgende Beispiel zeigt eine **`.audit`**-Datei, die den Warndienst prüft:

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

WMI_POLICY

Syntax

```
<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>
```

Mit diesem Test wird die Windows-WMI-Datenbank auf Werte abgefragt, die in dem entsprechenden Namespace, der Klasse oder dem Attribut angegeben sind.

Je nach verwendeter Syntax werden entweder Schlüsselwerte extrahiert oder Attributnamen aufgelistet.

Zulässige Typen sind:

```
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
```

Wenn Sie aus einer Dienstkonfiguration mit doppelten Werten auf dem System auswählen (z. B. „MSFTPSVC/83207416“ und „MSFTPSVC/2“), extrahiert die Anfrage das gewählte Attribut aus beiden. Entspricht eines davon nicht dem Richtlinienwert, dann wird der Wert von **wmi_key** dem Bericht hinzugefügt, um zu signalisieren, welcher Test nicht erfolgreich war. Mit dem Feld **wmi_enum** können Sie Konfigurationsnamen in einem Namespace zu Vergleichszwecken oder zum Testen des Richtlinienwertes auflisten.

Wenn eine WMI-Abfrage keine Ausgabe zurückgibt, meldet der Test standardmäßig einen Fehler. Dieses Verhalten kann geändert und der Test zur Meldung von PASS („Bestanden“) erzwungen werden, wenn **wmi_option** auf `CAN_BE_NULL` festgelegt wird. Wenn Sie **only_show_query_output** auf YES („Ja“) festlegen, wird die Ausgabe der WMI-Abfrage nachfolgend in den Nessus-Bericht eingefügt. Mit dem Tag **check_type** ist ein PASS-Ergebnis möglich, sofern ein bestimmter String nicht in der Ausgabe enthalten ist. Sehen Sie sich dazu die Beispiele weiter unten an.

Zusätzliche Anmerkungen:

- WMI-Attribute müssen explizit angegeben werden. Die Auswahl von `select * from foo` ist beispielsweise nicht möglich.
- Attribute ohne angegebenen Wert werden nicht gemeldet.
- Die Attribute sollten in ihrer Schreibweise genau so angegeben werden, wie sie in der Microsoft-Dokumentation erscheinen. Das Attribut `HandleCount` beispielsweise darf nicht als `Handlecount` oder `handlecount` angegeben werden.
- Werte des Arraytyps sind in den Ergebnissen nicht enthalten.

Beispiel 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>
```

Wenn auf Ihrem System zwei FTP-Dienstkonfigurationen vorhanden sind (z. B. „MSFTPSVC/83207416“ und „MSFTPSVC/2“), extrahiert die Anfrage das Attribut „UserIsolationMode“ aus beiden. Entspricht eines davon nicht dem Richtlinienwert (0), dann wird (in diesem Fall) der Wert von **wmi_key** dem Bericht hinzugefügt, um zu signalisieren, welcher Test nicht erfolgreich war.

Beispiel 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>
```

Dieses Beispiel überprüft, ob zwei gültige Konfigurationsnamen entsprechend der Angabe in **value_data** vorhanden sind. Wenn Sie mehr zum WMI-Namespace und zu zugehörigen Attributen erfahren möchten, setzen Sie Microsoft WMI CIM Studio ein. Dieses nützliche Tool kann von der folgenden Adresse heruntergeladen werden:

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

Beispiel 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>
```

Dieses Beispiel listet alle Windows-Prozesse auf; Instanzen von **svchost.exe** und **iPodService.exe** werden jedoch ausgelassen.

Items

„Items“ (Elemente) sind Testtypen, die in der Compliancetest-Engine für Windows vordefiniert sind. Sie werden für häufig geprüfte Elemente verwendet und reduzieren die für die Erstellung von Audittests erforderliche Syntax auf ein Minimum. Ein Element hat die folgende Struktur:

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

Das Feld **name** muss einen bereits definierten Namen enthalten (vordefinierte Namen sind unten in der Tabelle „Vordefinierte Richtlinien“ aufgeführt).

Alle vordefinierten Elemente entsprechen der Liste im Domänenrichtlinien-Editor von Windows 2003 SP1.

Im folgenden Beispiel wird geprüft, ob die minimale Kennwortlänge zwischen 8 und 14 Zeichen beträgt:

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

Das entsprechende benutzerdefinierte Element ist:

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

Vordefinierte Richtlinien

Richtlinie	Syntax

Password Policy („Kennwortrichtlinie“)	`name: "Enforce password history"`
	`value: POLICY_DWORD`
	
	`name: "Maximum password age"`
	`value: TIME_DAY`
	
	`name: "Minimum password age"`
	`value: TIME_DAY`
	
	`name: "Minimum password length"`
	`value: POLICY_DWORD`
	
	`name: "Password must meet complexity requirements"`
	`value: POLICY_SET`

Account Lockout Policy („Kontosperrungsrichtlinie“)	`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 Policy („Kerberos-Richtlinie“)	`name: "Maximum lifetime for service ticket"`
	`value: TIME_MINUTE`
	
	`name: "Maximum lifetime for user ticket"`
	`value: TIME_HOUR`
	
	`name: "Maximum lifetime for user renewal ticket"`
	`value: TIME_DAY`
	
	`name: "Maximum tolerance for computer clock synchronization"`

	value: TIME_MINUTE
Audit Policy („Auditrichtlinie“)	name: "Audit account logon events" value: AUDIT_SET name: "Audit account management" value: AUDIT_SET name: "Audit directory service access" value: AUDIT_SET name: "Audit logon events" value: AUDIT_SET name: "Audit object access" value: AUDIT_SET name: "Audit policy change" value: AUDIT_SET name: "Audit privilege use" value: AUDIT_SET name: "Audit process tracking" value: AUDIT_SET name: "Audit system events" value: AUDIT_SET
Accounts („Konten“)	name: "Accounts: Administrator account status" value: POLICY_SET name: "Accounts: Guest account status" value: POLICY_SET name: "Accounts: Limit local account use of blank password to console logon only" value: POLICY_SET name: "Accounts: Rename administrator account" value: POLICY_TEXT name: "Accounts: Rename guest account" value: POLICY_TEXT
Audit („Überwachung“)	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
Devices („Geräte“)	name: "Devices: Allow undock without having to log on" value: POLICY_SET name: "Devices: Allowed to format and eject removable media" value: DASD_SET name: "Devices: Prevent users from installing printer drivers" value: POLICY_SET name: "Devices: Restrict CD-ROM access to locally logged-on user only" value: POLICY_SET name: "Devices: Restrict floppy access to locally logged-on user only" value: POLICY_SET name: "Devices: Unsigned driver installation behavior" value: DRIVER_SET
Domain controller („Domänencontroller“)	name: "Domain controller: Allow server operators to schedule tasks" value: POLICY_SET name: "Domain controller: LDAP server signing requirements" value: LDAP_SET name: "Domain controller: Refuse machine account password changes" value: POLICY_SET
Domain member („Domänenmitglied“)	name: "Domain member: Digitally encrypt or sign secure channel data (always)" value: POLICY_SET name: "Domain member: Digitally encrypt secure channel data (when possible)" value: POLICY_SET name: "Domain member: Digitally sign secure channel data (when possible)" value: POLICY_SET name: "Domain member: Disable machine account password changes" value: POLICY_SET name: "Domain member: Maximum machine account password age" value: POLICY_DAY name: "Domain member: Require strong (Windows 2000 or later)

	session key" value: POLICY_SET
Interactive logon („Interaktive Anmeldung“)	name: "Interactive logon: Display user information when the session is locked" value: LOCKEDID_SET name: "Interactive logon: Do not display last user name" value: POLICY_SET name: "Interactive logon: Do not require CTRL+ALT+DEL" value: POLICY_SET name: "Interactive logon: Message text for users attempting to log on" value: POLICY_TEXT name: "Interactive logon: Message title for users attempting to log on" value: POLICY_TEXT name: "Interactive logon: Number of previous logons to cache (in case domain controller is not available)" value: POLICY_DWORD name: "Interactive logon: Prompt user to change password before expiration" value: POLICY_DWORD name: "Interactive logon: Require Domain Controller authentication to unlock workstation" value: POLICY_SET name: "Interactive logon: Require smart card" value: POLICY_SET name: "Interactive logon: Smart card removal behavior" value: SMARTCARD_SET
Microsoft network client („Microsoft-Netzwerk (Client)“)	name: "Microsoft network client: Digitally sign communications (always)" value: POLICY_SET name: "Microsoft network client: Digitally sign communications (if server agrees)" value: POLICY_SET name: "Microsoft network client: Send unencrypted password to third-party SMB servers" value: POLICY_SET
Microsoft network server („Microsoft-Netzwerk (Server)“)	name: "Microsoft network server: Amount of idle time required before suspending session" value: POLICY_DWORD name: "Microsoft network server: Digitally sign communications (always)"

	value: POLICY_SET name: "Microsoft network server: Digitally sign communications (if client agrees)" value: POLICY_SET name: "Microsoft network server: Disconnect clients when logon hours expire" value: POLICY_SET
Network access („Netzwerkzugriff“)	name: "Network access: Allow anonymous SID/Name translation" value: POLICY_SET name: "Network access: Do not allow anonymous enumeration of SAM accounts" value: POLICY_SET name: "Network access: Do not allow anonymous enumeration of SAM accounts and shares" value: POLICY_SET name: "Network access: Do not allow storage of credentials or .NET Passports for network authentication" value: POLICY_SET name: "Network access: Let Everyone permissions apply to anonymous users" value: POLICY_SET name: "Network access: Named Pipes that can be accessed anonymously" value: POLICY_MULTI_TEXT name: "Network access: Remotely accessible registry paths and sub-paths" value: POLICY_MULTI_TEXT name: "Network access: Remotely accessible registry paths" value: POLICY_MULTI_TEXT name: "Network access: Restrict anonymous access to Named Pipes and Shares" value: POLICY_SET name: "Network access: Shares that can be accessed anonymously" value: POLICY_MULTI_TEXT name: "Network access: Sharing and security model for local accounts" value: LOCALACCOUNT_SET
Network security („Netzwerksicherheit“)	name: "Network security: Do not store LAN Manager hash value on next password change" value: POLICY_SET name: "Network security: Force logoff when logon hours expire"

	value: POLICY_SET name: "Network security: LAN Manager authentication level" value: LANMAN_SET name: "Network security: LDAP client signing requirements" value: LDAPCLIENT_SET name: "Network security: Minimum session security for NTLM SSP based (including secure RPC) clients" value: NTLMSSP_SET name: "Network security: Minimum session security for NTLM SSP based (including secure RPC) servers" value: NTLMSSP_SET
Recovery console („Wiederherstellungskonsole“)	name: "Recovery console: Allow automatic administrative logon" value: POLICY_SET name: "Recovery console: Allow floppy copy and access to all drives and all folders" value: POLICY_SET
Shutdown („Herunterfahren“)	name: "Shutdown: Allow system to be shut down without having to log on" value: POLICY_SET name: "Shutdown: Clear virtual memory pagefile" value: POLICY_SET
System cryptography („Systemkryptographie“)	name: "System cryptography: Force strong key protection for user keys stored on the computer" value: CRYPTO_SET name: "System cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing" value: POLICY_SET
System objects („Systemobjekte“)	name: "System objects: Default owner for objects created by members of the Administrators group" value: OBJECT_SET name: "System objects: Require case insensitivity for non-Windows subsystems" value: POLICY_SET name: "System objects: Strengthen default permissions of internal system objects (e.g. Symbolic Links)" value: POLICY_SET
System settings („Systemeinstellungen“)	name: "System settings: Optional subsystems" value: POLICY_MULTI_TEXT name: "System settings: Use Certificate Rules on Windows Executables for Software Restriction Policies" value: POLICY_SET

Event Log („Ereignisprotokoll“)	`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`

Erzwungene Berichterstellung

Die Ausgabe eines bestimmten Ergebnisses durch Auditrichtlinien kann erzwungen werden. Hierzu wird das Schlüsselwort „**`report`**“ verwendet. Die Typen „PASSED“ („Bestanden“), „FAILED“ („Nicht bestanden“) und „WARNING“ („Warnung“) können eingesetzt werden. Das folgende Beispiel zeigt eine solche Richtlinie:

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

Der Text im Feld „**`description`**“ wird im Bericht immer angezeigt.

Diese Form der Berichterstellung ist nützlich, wenn Sie einen Prüfer darüber informieren möchten, dass ein Test, der gerade von Nessus ausgeführt wird, nicht abgeschlossen werden kann. Vielleicht besteht die Notwendigkeit, festzustellen, ob ein bestimmtes System physisch abgesichert ist, und Sie möchten den Prüfer darüber informieren, dass der Test oder die Überprüfung manuell ausgeführt werden muss. Diese Art des Berichts ist auch dann nützlich, wenn der Audittyp, der von Nessus ausgeführt werden soll, nicht mit einem OVAL-Test ermittelt wurde.

Bedingungen

In Windows-Richtlinien kann eine **if/then/else**-Logik definiert werden. Diese ermöglicht es, einen Test nur dann zu starten, wenn bestimmte Voraussetzungen erfüllt sind, oder mehrere Tests zu einem zusammenzufassen.

Die Syntax zur Ausführung von Bedingungen lautet wie folgt:

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

Bei den Bedingungen wird zwischen den Typen „**and**“ und „**or**“ unterschieden.

Das Audit der Bedingung der „**then**“- und „**else**“-Anweisungen kann eine Liste mit Elementen (oder benutzerdefinierten Elementen) oder eine „**if**“-Anweisung sein. Die „**else**“- und „**then**“-Anweisungen können zudem über den Typ „**report**“ je nach Rückgabewert der Bedingung einen Erfolg oder Fehlschlag melden:

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

Ein „**if**“-Wert gibt „SUCCESS“ („Erfolg“) oder „FAILURE“ („Fehlschlag“) zurück. Dieser Wert wird verwendet, wenn sich die „**if**“-Anweisung innerhalb einer anderen „**if**“-Struktur befindet. Wenn beispielsweise die **<then>**-Struktur ausgeführt wird, wird einer der folgenden Werte zurückgegeben:

- Das Audit enthält nur Elemente: Falls alle Elemente erfolgreich getestet wurden, wird „SUCCESS“ zurückgegeben, andernfalls „FAILURE“.
- Das Audit enthält nur **<report>**: Der Berichtstyp wird zurückgegeben.
- Das Audit enthält sowohl Elemente als auch **<report>**: Der Berichtstyp wird zurückgegeben.

Wenn die **<report>**-Anweisung verwendet wird und der Typ „FAILED“ ist, wird der Grund für den Fehlschlag – ggf. gemeinsam mit einem definierten Schweregrad – im Bericht aufgeführt.

Das folgende Beispiel prüft die Kennwortrichtlinie. Da der Typ „**and**“ verwendet wird, muss der Test beider benutzerdefinierter Elemente erfolgreich sein, damit die Überprüfung der Richtlinie ebenfalls erfolgreich ist. Im folgenden Beispiel wird eine sehr ungewöhnliche Kombination gültiger Kennwortchronikrichtlinien getestet, um zu veranschaulichen, wie eine ausgeklügelte Testlogik implementiert werden kann:

```
<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>
```

Im obigen Beispiel wurde nur der neue „**report**-Typ gezeigt. Die Struktur von **if**/**then**/**else** unterstützt jedoch auch die Durchführung zusätzlicher Audits in den „**else**“-Klauseln. Innerhalb einer Bedingung können zudem verschachtelte **if**/**then**/**else**-Klauseln verwendet werden. Nachfolgend gezeigt ist ein noch komplexeres Beispiel:

```
<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>
```

In diesem Beispiel wird geprüft, sofern das Administratorkonto nicht umbenannt wurde, ob das Kennwortmindestalter bei maximal 30 Tagen liegt. Diese Auditrichtlinie ist in jedem Fall erfolgreich, wenn das Administratorkonto umbenannt wurde; die Richtlinie zum Kennwortalter würde nur geprüft, wenn das Administratorkonto nicht umbenannt worden wäre.

Referenz zu Compliancedateien für Audits von Windows-Inhalten

`.audit`-Tests von Windows-Inhalten unterscheiden sich von **`.audit`**-Tests der Windows-Konfiguration dadurch, dass sie zum Durchsuchen eines Windows-Dateisystems nach bestimmten Dateitypen ausgelegt sind, die sensible Daten enthalten, statt Systemkonfigurationseinstellungen aufzulisten. Sie umfassen eine Anzahl von Optionen, die es dem Prüfer ermöglichen, die Suchparameter einzugrenzen und nichtkonforme Daten effektiver zu finden und anzuzeigen.

Verwendung von Anführungszeichen:

Um Auditfelder gesetzte einzelne bzw. doppelte Anführungszeichen werden austauschbar verwendet. Ausgenommen sind lediglich die folgenden Fälle:

1. Bei Windows-Compliancetests, in denen Sonderfelder (z. B. CRLF) literal interpretiert werden müssen, müssen einzelne Anführungszeichen gesetzt werden. Eingebettete Felder, die als Strings zu interpretieren sind, müssen mit Escapezeichen versehen werden.

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

2. Doppelte Anführungszeichen sind erforderlich, wenn „include_paths“ und „exclude_paths“ für WindowsFiles verwendet werden.

Wenn Strings in einem beliebigen Feldtyp (`description`, `value_data`, `regex` usw.) verwendet werden, die doppelte Anführungszeichen aufweisen, kann auf zweierlei Weise verfahren werden:

a. Verwenden Sie den jeweils anderen Anführungszeichentyp als äußere umschließende Anführungszeichen.

Beispiel:

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

b. Stellen Sie eingebetteten doppelten Anführungszeichen den umgekehrten Schrägstrich („\“) als Escapezeichen voran.

Beispiel:

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

Testtyp

Alle Windows-Inhaltscompliancetests müssen in die Kapselung **`check_type`** gesetzt werden, die mit der Typangabe „**`WindowsFiles`**“ zu versehen ist. Dies ist bei allen anderen **`.audit`**-Dateien sehr ähnlich. Das grundlegende Format eines Inhaltstests sieht wie folgt aus:

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

Die eigentlichen Tests für einzelne Elemente werden nicht angezeigt. Die folgenden Abschnitte zeigen, wie mithilfe verschiedener Schlüsselwörter und Parameter ein bestimmtes Inhaltselementaudit gefüllt werden kann.

Elementformat

Syntax
<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>

Die folgenden Elemente werden für Audits einer Vielzahl von Dateiformaten und mit den unterschiedlichsten Datentypen verwendet. Die nachfolgende Tabelle enthält eine Liste der unterstützten Datentypen. Im nächsten Abschnitt finden Sie zahlreiche Beispiele zur gemeinsamen Verwendung dieser Schlüsselwörter für Audits verschiedener Dateiinhaltstypen.

Schlüsselwort	Beschreibung
`type`	Muss immer auf „FILE_CONTENT_CHECK“ festgelegt sein.
`description`	Diese Angabe wird als Titel für eindeutige Compliancesicherheitslücken in SecurityCenter verwendet. Außerdem ist dies der erste von Nessus gemeldete

	Datensatz.
file_extension	Hier sind alle Erweiterungen aufgeführt, nach denen Nessus suchen soll. Die Erweiterungen werden ohne den vorangestellten Punkt („.“), dafür in Anführungszeichen und durch Pipes getrennt aufgeführt. Sind weitere Optionen wie **regex** und **expect** nicht im Audit enthalten, dann werden Dateien mit der hier gewählten Erweiterung in der Auditausgabe aufgeführt.
regex	Dieses Schlüsselwort enthält den regulären Ausdruck, der für die Suche nach komplexen Datentypen verwendet wird. Liegt eine Übereinstimmung mit dem regulären Ausdruck vor, dann wird der erste übereinstimmende Inhalt im Sicherheitslückenbericht angezeigt.  Das Schlüsselwort **regex** muss gemeinsam mit dem nachfolgend beschriebenen Schlüsselwort **expect** ausgeführt werden.  Anders als bei Compliancetests des Typs „Windows“ müssen bei Inhaltstests für Windows-Dateien **regex** und **expect** nicht mit denselben Datenstrings in der durchsuchten Datei übereinstimmen. Hier ist es lediglich erforderlich, dass sowohl **regex**- als auch **expect**-Anweisungen mit den Daten im Bereich der `<max_size>` Bytes der durchsuchten Datei übereinstimmen.
expect	Mithilfe der **expect**-Anweisung werden ein oder mehrere einfache Muster aufgelistet, die im Dokument vorhanden sein müssen, damit eine Übereinstimmung vorliegt. Bei der Suche nach US-amerikanischen Sozialversicherungsnummern beispielsweise könnten dies die Muster „SSN“, „SS#“ oder „Social“ sein. Mehrere Muster werden jeweils in Anführungszeichen gesetzt und durch Pipes voneinander getrennt. Ein einfacher Mustervergleich unter Verwendung des Punktes wird ebenfalls unterstützt. Wenn Sie beispielsweise nach dem String „C.T“ suchen, liegt für die **expect**-Anweisung eine Übereinstimmung bei Zeichenfolgen wie „CAT“, „CaT“, „COT“, „C T“ usw. vor.  Das Schlüsselwort **expect** kann wahlweise alleine für einen Vergleich mit nur einem Muster verwendet werden; wenn jedoch das Schlüsselwort **regex** verwendet wird, ist **expect** erforderlich.  Anders als bei Windows-Compliancetests müssen **regex** und **expect** beim Compliancetest für Windows-Dateiinhalte nicht mit denselben Datenstrings in der durchsuchten Datei übereinstimmen. Hier ist es lediglich erforderlich, dass sowohl **regex**- als auch **expect**-Anweisungen mit den Daten im Bereich der `<max_size>` Bytes der durchsuchten Datei übereinstimmen.

file_name	Zwar ist das Schlüsselwort **file_extension** obligatorisch, doch lässt sich hiermit die Liste der zu analysierenden Dateien weiter eingrenzen. Durch Angabe einer Liste mit Mustern können Dateien verworfen oder verglichen werden. Dies macht es beispielsweise sehr einfach, nach beliebigen Dateinamenstypen zu suchen, in denen etwa die Zeichenfolgen „Mitarbeiter“, „Kunde“ oder „Gehalt“ enthalten sind.
max_size	Aus Gründen der Leistungsoptimierung kann das Audit auf den ersten Teil jeder Datei beschränkt werden. Die Größe dieses ersten Teils in Byte kann mit diesem Schlüsselwort angegeben werden. Die Anzahl der Bytes kann dabei als Argument verwendet werden. Ebenfalls werden die Erweiterungen „K“ und „M“ für Kilobyte bzw. Megabyte unterstützt.
only_show	Beim Vergleich sensibler Daten wie Kreditkartennummern sieht Ihr Unternehmen möglicherweise vor, dass nur die letzten vier Ziffern im Bericht aufgeführt werden. Dieses Schlüsselwort ermöglicht die Anzeige einer beliebigen Anzahl von Bytes, wie sie durch die Richtlinie festgelegt ist.
regex_replace	Mit diesem Schlüsselwort wird gesteuert, welches Muster im regulären Ausdruck im Bericht angezeigt wird. Bei der Suche nach komplexen Datenmustern wie etwa Kreditkartennummern ist es nicht immer möglich, sicherzustellen, dass die erste Übereinstimmung den gesuchten Daten entspricht. Dieses Schlüsselwort bietet bei der Erfassung der gewünschten Daten mehr Flexibilität bei größerer Genauigkeit.
include_paths	Dieses Schlüsselwort ermöglicht die Aufnahme von Verzeichnis- oder Laufwerksangaben in die Suchergebnisse. Das Schlüsselwort wird in Verbindung mit oder unabhängig vom Schlüsselwort „**exclude_paths**“ verwendet. Es ist besonders nützlich in Fällen, in denen nur bestimmte Laufwerke oder Ordner auf einem System mit mehreren Laufwerken durchsucht werden sollen. Wenn mehrere Pfade erforderlich sind, werden diese in doppelte Anführungszeichen gesetzt und durch das Pipe-Symbol voneinander getrennt.  Als Schlüsselwort „**include_paths**“ können nur Laufwerksbuchstaben und Ordnernamen angegeben werden. Dateinamen können nicht in den „**include_paths**“-String gesetzt werden.
exclude_paths	Dieses Schlüsselwort ermöglicht den Ausschluss von Laufwerken, Verzeichnissen oder Dateien aus den Suchergebnissen. Das Schlüsselwort wird entweder in Verbindung mit oder unabhängig vom Schlüsselwort „**include_paths**“ verwendet. Es ist besonders nützlich in Fällen, in denen bestimmte Laufwerke, Verzeichnisse oder Dateien aus den Suchergebnissen ausgeschlossen werden sollen. Wenn mehrere Pfade erforderlich sind, werden diese in doppelte Anführungszeichen gesetzt und durch das Pipe-Symbol voneinander getrennt.

see_also	Dieses Schlüsselwort lässt die Angabe von Verweisen auf Referenzmaterial zu.
	
	Beispiel:
	`see_also:`
	`"https://benchmarks.cisecurity.org/tools2/linux/CIS_Redhat_Linux_5_Benchmark_v2.0.0.pdf"`

solution	Dieses Schlüsselwort ermöglich den Zusatz eines Lösungstexts, sofern vorhanden. Beispiel: `solution : "Remove this file, if its not required"`

reference	Mit diesem Schlüsselwort können Sie Querverweise in die **.audit**-Datei setzen. Das Format ist „ref|ref-id1,ref|ref-id2“. Beispiel: `reference : "CAT|CAT II,800-53|IA-5,8500.2|IAIA-1,8500.2|IAIA-2,8500.2|IATS-1,8500.2|IATS-2"`

Beispiele für die Befehlszeile

In diesem Abschnitt erstellen wir ein fiktives Textdokument mit der Erweiterung **.tns** und lassen es dann durch einfache wie auch komplexe **.audit**-Dateien laufen. Bei der Beschreibung der einzelnen Beispiele werden wir alle unterstützten Parameter für Windows-Inhalte ausprobieren.

Außerdem werden wir die **nasl**-Befehlszeilenbinärdatei verwenden. Alle gezeigten **.audit**-Dateien können Sie auch auf Ihre Nessus 4 oder SecurityCenter-Scanrichtlinien ziehen, doch für schnelle Audits eines einzelnen Systems ist diese Vorgehensweise sehr effizient. Den folgenden Befehl werden wir jedes Mal aus dem Verzeichnis **/opt/nessus/bin** heraus ausführen:

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

<IP> ist die IP-Adresse des geprüften Systems.

Wenn Sie bei Nessus 4 die **.nbin**-Datei (oder ein anderes Plugin) ausführen, werden Sie zur Eingabe der Anmeldedaten für das Zielsystem sowie des Speicherorts der **.audit**-Datei aufgefordert.

Zieltestdatei

Die von uns verwendete Zieldatei hat den folgenden Inhalt:

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

Kopieren Sie diese Daten auf ein Windows-System, auf das Sie authentifizierten Zugriff haben. Nennen Sie die Datei „Tenable_Content.tns“.

Beispiel 1: Nach .tns-Dokumenten suchen, die das Wort „Nessus“ enthalten

Nachfolgend gezeigt ist eine einfache **.audit**-Datei, die nach **.tns**-Dateien sucht, in denen an irgendeiner Stelle das Wort „Nessus“ enthalten ist.

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

```
</check_type>
```

Wenn Sie diesen Befehl ausführen, müsste die folgende Ausgabe erscheinen:

```
"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
```

Den Ergebnissen lässt sich entnehmen, dass eine Übereinstimmung gefunden wurde. Im Bericht erscheint die Meldung „failed“ („Nicht bestanden“), weil Daten gefunden wurden, nach denen wir nicht gesucht haben. Wenn Sie beispielsweise eine Prüfung auf Sozialversicherungsnummern durchführen und eine solche Nummer auf einem öffentlich zugänglichen Computer gefunden wird, haben wir zwar ein positives Ergebnis erhalten, doch wird es als „failed“ protokolliert, weil infolge dieses Ergebnisses die Compliance-Anforderungen nicht erfüllt wurden.

Beispiel 2: Nach `.tns`-Dokumenten suchen, die das Wort „France“ enthalten

Nachfolgend gezeigt ist eine einfache **`.audit`**-Datei, die nach **`.tns`**-Dateien sucht, in denen an irgendeiner Stelle das Wort „France“ enthalten ist.

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

Diesmal sieht die Ausgabe wie folgt aus:

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

Das Audit wurde als erfolgreich („pass“) gewertet, weil das Wort „France“ in keiner der untersuchten **`.tns`**-Dateien vorhanden war.

Beispiel 3: Nach `.tns`- und `.doc`-Dokumenten suchen, die das Wort „Nessus“ enthalten

Das Hinzufügen einer zweiten Erweiterung für die Suche nach Microsoft Word-Dokumenten ist sehr einfach:

```
<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>
```

Auf unseren Testcomputern erhielten wir folgende Ergebnisse:

```
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
```

Auch hier erscheint wie zuvor die Meldung „failed“ für unsere **.tns**-Testdatei, doch gab es in diesem Fall eine zweite Datei – nämlich eine **.doc**-Datei –, die das Wort „Nessus“ enthielt. Wenn Sie diese Tests auf Ihren eigenen Systemen ausführen, finden Sie eventuell, aber nicht unbedingt eine Word-Datei, die das Wort „Nessus“ enthält.

Beispiel 4: Nach .tns- und .doc-Dokumenten suchen, die das Wort „Nessus“ und eine elfstellige Zahl enthalten

Nun fügen wir unseren ersten regulären Ausdruck hinzu: die Überprüfung auf Vorhandensein einer elfstelligen Zahl. Wir müssen hierzu der oben beschriebenen **.audit**-Datei einfach nur den regulären Ausdruck mit dem Schlüsselwort **regex** hinzufügen.

```
<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>
```

Bei der Ausführung wird folgende Ausgabe erzeugt:

```
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)
```

Die im vorangegangenen Beispiel gefundene **.doc**-Datei wird erneut durchsucht. Da sie keine elfstellige Zahl enthält, erscheint sie jedoch nicht mehr in den Ergebnissen. Beachten Sie auch, dass aufgrund der Verwendung des Schlüsselworts **regex** ein Ergebnis in den Daten angezeigt wird.

Nun wollen wir einmal annehmen, dass wir nach einer zehnstelligen Zahl suchen. Die obige elfstellige Zahl enthält zwei zehnstellige Zahlen (nämlich 0123456789 und 1234567890). Wenn wir zur genaueren Übereinstimmung elf Stellen schreiben möchten, benötigen wir eigentlich einen regulären Ausdruck wie den folgenden:

„Suche nach einer beliebigen elfstelligen Zahl, der keine weiteren Ziffern voran- oder nachgestellt sind.“

Zu diesem Zweck können wir in regulären Ausdrücken den Operator „not“ („nicht“) wie folgt verwenden:

```
<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>
```

Beim Lesen des Codes stößt man auf die Zeichen „^“ und „$“. Das Zeichen „^“ hat entweder die Bedeutung eines Zeilenanfangs oder führt einen Vergleich mit dem negativen Wert des Nachgestellten aus. Das Dollarzeichen bezeichnet ein Zeilenende. Der obige reguläre Ausdruck besagt im Grunde genommen, dass nach beliebigen Mustern gesucht werden soll, die nicht mit einer Zahl beginnen, möglicherweise aber am Zeilenanfang stehen und elf Ziffern enthalten, denen keine weiteren Ziffern nachgestellt sind, die aber am Zeilenende stehen können. Reguläre Ausdrücke behandeln Zeilenanfang und -ende als Sonderfälle, weswegen die Verwendung der Zeichen „^“ und „$“ erforderlich sein kann.

Beispiel 5: Nach `.tns` - und `.doc`-Dokumenten suchen, die das Wort „Nessus“ und eine elfstellige Zahl enthalten, und von dieser Zahl nur die letzten vier Ziffern anzeigen

Durch Hinzufügen des Schlüsselworts **`only_show`** zu unserer **`.audit`**-Datei können wir die Ausgabe beschränken. Auf diese Weise ist es möglich, Prüfern nur Zugang zu solchen sensiblen Daten zu gewähren, die sie tatsächlich benötigen.

```
<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>
```

Die Ziffern gefundener Werte werden wie nachfolgend gezeigt teilweise durch X-Zeichen ersetzt:

```
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)
```

Beispiel 6: Nach .tns-Dokumenten suchen, die das Wort „Correlation“ in den ersten 50 Bytes enthalten

In diesem Beispiel werden wir die Verwendung des Schlüsselwortes **`max_size`** kennen lernen. In unserer Testdatei ist das Wort „Correlation“ zwar enthalten, jedoch nicht in den ersten 50 Bytes.

```
<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>
```

Bei der Ausführung erhalten wir das Ergebnis „passed“:

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

Wenn wir nun den Wert **`max_size`** von „50“ auf „50K“ setzen und den Scan erneut ausführen, erhalten wir eine Fehlermeldung:

```
"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
```

Beispiel 7: Steuern, was in der Ausgabe angezeigt wird

In diesem Beispiel werden wir die Verwendung des Schlüsselwortes **`regex_replace`** kennen lernen. Sehen Sie sich folgende **`.audit`**-Datei an:

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

Dieser Test erzeugt folgende Ausgabe:

```
"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)
```

Beachten Sie jedoch, was passieren kann, wenn wir tatsächlich einen regulären Ausdruck benötigen, der auf Vorhandensein der Teile „Passive“ und „Scanner“ prüft, aber nur den Teil „Vulnerability“ zurückgeben soll. Ein neuer regulärer Ausdruck würde wie folgt aussehen:

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

Doch auch dieser Test gibt das vollständige Ergebnis „Passive Vulnerability Scanner“ zurück, weil die reguläre Ausdrucksanweisung den gesamten String als erstes Ergebnis behandelt; wollen wir nur das zweite Ergebnis, dann müssen wir das Schlüsselwort **`regex_replace`** hinzufügen.

```
<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>
```

Die Scanausgabe sieht wie folgt aus:

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

Um festzulegen, dass wir das zweite Element des Suchergebnisses zurückgeben möchten, verwenden wir „\3“. Bei „\1“ wäre der gesamte String zurückgegeben worden, bei „\2“ das Wort „Passive“ und bei „\4“ das Wort „Scanner“.

Warum ist diese Funktion vorhanden? Bei der Suche nach komplexen Datenmustern wie etwa Kreditkartennummern ist es nicht immer möglich, sicherzustellen, dass die erste Übereinstimmung den gesuchten Daten entspricht. Dieses Schlüsselwort bietet bei der Erfassung der gewünschten Daten mehr Flexibilität und größere Genauigkeit.

Beispiel 8: Dateinamen als Filter verwenden

Gehen wir noch einmal zur **`.audit`**-Datei aus dem dritten Beispiel zurück. Dort wurde sowohl für eine **`.tns`**- als auch für eine **`.doc`**-Datei ein Ergebnis zurückgegeben.

```
<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>
```

Auf unseren Testcomputern erhielten wir folgende Ergebnisse:

```
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
```

Das Schlüsselwort **`file_name`** kann auch zum Ausfiltern von Dateien verwendet werden. Wenn Sie es zur **`.audit`**-Datei hinzufügen, können Sie den Vorgang wie folgt auf Dateien beschränken, in deren Name der String „tenable“ enthalten ist:

```
<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>
```

Die Ausgabe sieht wie folgt aus:

```
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
```

Die übereinstimmende **.doc**-Datei wird nicht ausgegeben, weil das Wort „tenable“ in ihrem Pfad nicht vorhanden ist.

Der Vergleichsstring ist ein regulärer Ausdruck, d. h., er kann sehr flexibel an eine Vielzahl von Dateien angepasst werden, die wir prüfen (oder nicht prüfen) möchten. Wir können mit dem String „[Tt]enable“ beispielsweise auf Vorhanden sein von „Tenable“ oder „tenable“ prüfen. Analog müssen wir, wenn wir eine Dateierweiterung oder einen Teil davon suchen, den Punkt mit einem Escapezeichen („\“) versehen. So sucht „\.t“ nach allen Erweiterungen, die mit „t“ beginnen.

Beispiel 9: Schlüsselwörter zum Ein- und Ausschließen verwenden

Die Schlüsselwörter „**include_paths**“ und „**exclude_paths**“ können zur Filterung von Suchvorgängen auf der Basis von Laufwerksbuchstabe oder Verzeichnis und sogar für einen Dateinamensausschluss verwendet werden.

```
<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>
```

Die Ausgabe sieht wie folgt aus:

```
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
```

Beachten Sie, dass die Ausgabe sich nicht vom Ergebnis einer normalen Suche nach Windows-Dateiinhalten unterscheidet, nur ist hier der Pfad ausgeschlossen. Wenn ein einzelner Pfad mit „**include_paths**“ eingeschlossen wird (z. B. „**c:**“), werden alle anderen Pfade automatisch ausgeschlossen. Weiterhin hat, wenn ein Laufwerksbuchstabe ausgeschlossen wird (z. B. „**d:**“), ein Ordner auf diesem Laufwerk jedoch eingeschlossen wird (z. B. „**d:\users**“), das Schlüsselwort „**exclude_paths**“ Vorrang, d. h., das Laufwerk wird nicht durchsucht. Sie können jedoch ein Laufwerk **C:** ein- und dann einen Ordner auf diesem Laufwerk (z. B. **C:\users:**) ausschließen.

Verschiedene Dateiformattypen prüfen

Audits lassen sich für alle Dateierweiterungen ausführen. Allerdings werden Formate wie **.zip** und **.gz** während des Vorgangs nicht dekomprimiert. Wenn Ihre Datei komprimiert ist oder die enthaltenen Daten auf irgendeine Weise kodiert wurden, ist eine Mustersuche nicht möglich.

Bei Dokumenten, die Daten im Unicode-Format speichern, werden alle gefundenen NULL-Bytes von den Parserroutinen der **.nbin**-Datei verworfen.

Außerdem werden Microsoft Office-Dokumente aus allen Versionen unterstützt. Dies gilt auch für die neuen, seit Office 2007 verwendeten Formate wie **.xlsx** und **.docx**.

Schließlich werden auch verschiedene PDF-Dateiformate unterstützt. Tenable hat ein umfangreiches PDF-Analyseprogramm entworfen, das für den Vergleich die reinen Textstrings extrahiert. Deswegen müssen Benutzer sich lediglich die Frage stellen, nach welchen Daten sie in einer PDF-Datei suchen möchten.

Leistungsaspekte

Es gibt eine Reihe von Kompromissen, die eingegangen werden müssen, wenn man die **.audit**-Standarddateien ändert und sie in Netzwerken testet, die online sind:

- Nach welchen Erweiterungen soll gesucht werden?
- Wie viele Daten müssen gescannt werden?

Die **.audit**-Dateien erfordern das Schlüsselwort **max_size** nicht. In diesem Fall versucht Nessus, die gesamte Datei abzurufen, und verarbeitet diese so lange, bis ein Ergebnis für ein Muster gefunden wurde. Da diese Dateien das gesamte Netzwerk durchlaufen, gibt es bei solchen Audits mehr Datenverkehr als beim normalen Scannen oder bei Konfigurationsaudits.

Werden durch ein SecurityCenter mehrere Nessus-Scanner verwaltet, dann müssen die Daten nur vom gescannten Windows-Host bis zu dem Scanner übertragen werden, der das Sicherheitslückenaudit ausführt.

Referenz zu Compliancedateien für Cisco IOS-Konfigurationsaudits

In diesem Abschnitt werden Format und Funktionen von Cisco IOS-Compliancetests und das Prinzip hinter den einzelnen Einstellungen erläutert.

Verwendung von Anführungszeichen:

Um Auditfelder gesetzte einzelne bzw. doppelte Anführungszeichen werden austauschbar verwendet. Ausgenommen sind lediglich die folgenden Fälle:

1. Bei Windows-Compliancetests, in denen Sonderfelder (z. B. CRLF) literal interpretiert werden müssen, müssen einzelne Anführungszeichen gesetzt werden. Eingebettete Felder, die als Strings zu interpretieren sind, müssen mit Escapezeichen versehen werden.

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

2. Doppelte Anführungszeichen sind erforderlich, wenn „include_paths“ und „exclude_paths“ für WindowsFiles verwendet werden.

Wenn Strings in einem beliebigen Feldtyp (`description`, `value_data`, `regex` usw.) verwendet werden, die doppelte Anführungszeichen aufweisen, kann auf zweierlei Weise verfahren werden:

a. Verwenden Sie den jeweils anderen Anführungszeichentyp als äußere umschließende Anführungszeichen.

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

b. Stellen Sie eingebetteten doppelten Anführungszeichen den umgekehrten Schrägstrich („\“) als Escapezeichen voran.

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

Testtyp

Alle Cisco IOS-Compliancetests müssen in die Kapselung **check_type** gesetzt werden, die mit der Typangabe „Cisco“ zu versehen ist. Dies ist erforderlich, um **.audit**-Dateien für Systeme, die unter Cisco IOS laufen, von anderen Compliancetests unterscheiden zu können.

Beispiel:

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

Anders als bei anderen Compliance-Audittypen sind keine weiteren Schlüsselwörter zur Angabe von Typ oder Version vorhanden.

Schlüsselwörter

Die folgende Tabelle gibt an, wie die einzelnen Schlüsselwörter in Cisco-Compliancetests verwendet werden können:

Schlüsselwort	Einsatzbeispiel und unterstützte Einstellungen
type	CONFIG_CHECK, CONFIG_CHECK_NOT und RANDOMNESS_CHECK “Mit „CONFIG_CHECK“ wird geprüft, ob das angegebene Element in der Ausgabe des Cisco IOS-Befehls „show config“ vorhanden ist. Analog prüft „CONFIG_CHECK_NOT“, ob das angegebene Element nicht vorhanden ist. „RANDOMNESS_CHECK“ schließlich wird zur Überprüfung von Stringkomplexitätstests (z. B. Kennwortüberprüfungen) benutzt. Wenn Sie über einen regulären Ausdruck ein zu suchendes Element angeben, werden Sie hiermit darüber informiert, ob der Grad der Zufälligkeit hoch genug ist (d. h. ob das Wort mindestens acht Zeichen lang ist, eine gemischte Groß-/Kleinschreibung aufweist und jeweils mindestens eine Ziffer und ein Sonderzeichen enthält).  Die Zufälligkeitsparameter können gegenwärtig nicht konfiguriert werden.
description	Das Schlüsselwort „**description**“ bietet die Möglichkeit, dem ausgeführten Test eine kurze Beschreibung hinzuzufügen. Es wird dringend empfohlen, dem Feld **description** einen eindeutigen Wert zuzuweisen und sicherzustellen, dass andere Tests keinen identischen Eintrag im Feld **description** aufweisen. Auf der Basis des Feldes **description** generiert Tenable SecurityCenter automatisch eine eindeutige Plugin-ID. Beispiel: `description: "Forbid Remote Startup Configuration"`

feature_set	Das Schlüsselwort „**feature_set**“ wird ähnlich wie das Schlüsselwort „**system**“ in UNIX-Compliancetests verwendet, um die Featureset-Version von Cisco IOS zu testen. Es führt den resultierenden Test entweder aus oder übergeht ihn aufgrund eines unzutreffenden regulären Ausdrucks. Dies ist praktisch in Fällen, in denen ein Test nur auf Systemen mit einem bestimmten Featureset ausgeführt werden soll.
	
	Beispiel:
	`<item>`
	`type: CONFIG_CHECK`
	`description: "Version Check"`
	`info: "SSH Access Control Check."`
	`feature_set: "K8" context:"line .*"`
	`item: "access-class [0-9]+ in"`
	`</item>`
	
	Der obige Test führt den „item“-Test nur aus, wenn die Featureset-Version dem angegebenen regulären Ausdruck („K8“) entspricht.
	
	Im Falle eines fehlgeschlagenen Featureset-Versionstests wird ein Fehler ähnlich dem nachfolgenden angezeigt:
	
	`"Version Check" : [SKIPPED]`
	`Test defined for the feature set 'K8' whereas we are running C850-ADVSECURITYK9-M`

ios_version	Das Schlüsselwort „**ios_version**“ wird ähnlich wie das Schlüsselwort „**system**“ in UNIX-Compliancetests verwendet, um die Version von Cisco IOS zu testen. Es führt den resultierenden Test entweder aus oder übergeht ihn aufgrund eines unzutreffenden regulären Ausdrucks. Dies ist praktisch in Fällen, in denen ein Test nur auf Systemen mit einer bestimmten IOS-Version ausgeführt werden soll.
	
	Beispiel:
	`<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>`
	
	Der obige Test führt den „item“-Test nur aus, wenn die IOS-Version dem angegebenen regulären Ausdruck („12\.[5-9]“) entspricht.
	
	Im Falle eines fehlgeschlagenen IOS-Versionstests wird ein Fehler ähnlich dem nachfolgenden angezeigt:
	
	`"Version Check" : [SKIPPED]`
	`Test defined for 12.[5-9] whereas we are running 12.4(15)T10`

info	Das Schlüsselwort „**info**“ wird verwendet, um dem ausgeführten Test eine ausführlichere Beschreibung hinzuzufügen. Dies könnte ein Verweis auf eine Rechtsvorschrift, eine URL mit weiteren Informationen, eine Unternehmensrichtlinie usw. sein. Mehrere **info**-Felder lassen sich in separaten Zeilen hinzufügen, um den Text als Abschnitt zu formatieren. Die Anzahl der verwendbaren **info**-Felder ist nicht begrenzt.

	
	Jedes „**info**“-Tag muss in eine einzelne Zeile ohne Zeilenumbruch geschrieben werden. Wenn mehrere Zeilen erforderlich sind (z. B. zu Formatierungszwecken), fügen Sie einfach weitere „**info**“-Tags hinzu.
	
	Beispiel:
	`info: "Verify at least one local user exists and ensure"`
	`info: "all locally defined user passwords are protected"`
	`info: "by encryption."`

item	Das Schlüsselwort „**item**“ gibt das Konfigurationselement in der zu prüfenden Ausgabe des Befehls „show config“ an. Beispiel: `item: "transport input ssh"` Mithilfe regulärer Ausdrücke, die in diesem Schlüsselwort verwendet werden, können die Ergebnisse des Vergleichs gefiltert werden. Weitere Informationen zur Funktionalität von **regex** entnehmen Sie der Beschreibung zum Schlüsselwort **regex**.
regex	Mit dem Schlüsselwort „**regex**“ aktivieren Sie die Suche nach der Konfigurationselementeinstellung, die einem bestimmten regulären Ausdruck entspricht. Beispiel: `regex: "snmp-server community ([^ ]*) .*"` Die folgenden Metazeichen erfordern einen gesonderten Umgang: + \ * ( ) ^ Wenn diese Zeichen literal ausgewertet werden sollen, müssen Sie sie entweder mit zwei umgekehrten Schrägstrichen („\\“) als Escapezeichen auszeichnen oder sie in eckige Klammern („[]“) setzen. Die folgenden Zeichen hingegen müssen für eine literale Auswertung mit nur einem umgekehrten Schrägstrich ausgezeichnet werden: . ? " '. Dies hat mit der Art und Weise zu tun, wie der Compiler diese Zeichen auswertet.
min_occurrences	Mit dem Schlüsselwort „**min_occurrences**“ geben Sie an, wie häufig das Konfigurationselement mindestens auftreten muss, damit das Audit als erfolgreich gewertet wird. Beispiel: `min_occurrences: "3"`
max_occurrences	Mit dem Schlüsselwort „**max_occurrences**“ geben Sie an, wie häufig das Konfigurationselement höchstens auftreten darf, damit das Audit als erfolgreich gewertet wird. Beispiel: `max_occurrences: "1"`
required	Mit dem Schlüsselwort „**required**“ wird angegeben, ob das geprüfte Element auf dem Remotesystem vorhanden sein muss oder nicht. Wenn beispielsweise **required** den Wert „NO“ hat und als Testtyp „CONFIG_CHECK“ festgelegt wurde, ist der Test erfolgreich, wenn das Konfigurationselement vorhanden ist oder nicht vorhanden ist.

	Hätte **required** hingegen den Wert „YES“, dann würde der obige Test fehlschlagen. Beispiel: `required: NO`

context	Das Schlüsselwort „**context**“ ist nützlich, wenn mehrere Instanzen eines bestimmten Konfigurationselements vorhanden sind. Sehen Sie sich folgende Konfiguration an:
	
	`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`
	
	Wenn Sie einen Wert über eine bestimmte serielle Verbindung testen möchten, ist die Verwendung des Schlüsselworts **item** mit „line“ nicht ausreichend, da es für „line“ mehrere mögliche Optionen gibt. Verwenden Sie hingegen „**context**“, um das für Sie interessante Element anzugeben. Hier ein Beispiel:
	
	`context: "con 0"`
	
	Nun wird nur nach dem folgenden Konfigurationselement gesucht:
	
	`line con 0`
	` no modem enable`
	
	Mithilfe regulärer Ausdrücke, die in diesem Schlüsselwort verwendet werden, können die Ergebnisse des Vergleichs gefiltert werden. Weitere Informationen zur Funktionalität von **regex** entnehmen Sie der Beschreibung zum Schlüsselwort **regex**.

Beispiele für die Befehlszeile

In diesem Abschnitt sind mehrere Beispiele gängiger Audits enthalten, die in Cisco IOS-Compliancetests verwendet werden. Das Befehlszeilenbinärprogramm **nasl** wird als Möglichkeit verwendet, Tests zwischendurch schnell und unkompliziert durchzuführen. Sie können alle nachfolgend gezeigten **.audit**-Dateien problemlos in Ihre Nessus-Scanrichtlinien integrieren. Allerdings sind für schnelle Audits eines einzelnen Systems Befehlszeilentests effizienter. Der Befehl wird jedes Mal wie folgt aus dem Verzeichnis **/opt/nessus/bin** heraus ausgeführt:

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

<IP> ist die IP-Adresse des geprüften Systems.

Nun wird das Enable-Kennwort abgefragt:

```
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 :
```

Wenden Sie sich an Ihren Cisco-Administrator, um die richtigen Anmeldeparameter für den Enable-Zugriff zu erhalten.

Beispiel 1: Nach einer definierten SNMP-ACL suchen

Die nachfolgend gezeigte einfache **`.audit`**-Datei sucht nach einer definierten „deny“-ACL für SNMP. Wird eine solche nicht gefunden, dann wird gemeldet, dass das Audit fehlgeschlagen ist. Dieser Test wird nur ausgeführt, wenn die IOS-Version des Routers dem angegebenen regulären Ausdruck entspricht. Andernfalls wird der Test übersprungen.

```
<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>
```

Wenn Sie diesen Befehl ausführen, müsste auf einem System, das die Compliance-Anforderungen erfüllt, folgende Ausgabe erscheinen:

```
"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.
```

Bei einem fehlgeschlagenen Audit würde die folgende Ausgabe zurückgegeben:

```
"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
```

In diesem Fall schlug der Test fehl, weil nach einer „deny ip“-Regel gesucht wurde, die nicht gefunden wurde.

Beispiel 2: Sicherstellen, dass der `finger`-Dienst deaktiviert ist

Die nachfolgend gezeigte einfache **`.audit`**-Datei sucht nach dem unsicheren **`finger`**-Dienst auf dem Remoterouter. Dieser Test wird nur ausgeführt, wenn die IOS-Version des Routers dem angegebenen regulären Ausdruck entspricht. Andernfalls wird der Test übersprungen. Wird der Dienst gefunden, dann wird gemeldet, dass das Audit fehlgeschlagen ist.

```
<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>
```

Wenn Sie diesen Befehl ausführen, müsste auf einem System, das die Compliance-Anforderungen erfüllt, folgende Ausgabe erscheinen:

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

Disable finger server.
```

Bei einem fehlgeschlagenen Audit würde die folgende Ausgabe zurückgegeben:

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

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

Beispiel 3: SNMP-Community-Strings und Zugriffssteuerung auf ein ausreichendes Maß an Zufälligkeit prüfen

Die nachfolgend gezeigte einfache **`.audit`**-Datei sucht nach SNMP-Community-Strings, deren Grad an Zufälligkeit nicht ausreichend ist. Wenn ein Community-String gefunden wird, bei dem ein nicht ausreichender Grad an Zufälligkeit erkannt wird, zeigt das Audit an, dass der Test nicht erfolgreich war. Da die Option „required“ den Wert „NO“ hat, ist der Test auch dann erfolgreich, wenn keine SNMP-Community-Strings vorhanden sind. Der Test kann nur ausgeführt werden, wenn der Router das Featureset „K9“ verwendet. Andernfalls wird der Test übersprungen.

```
<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>
```

Wenn Sie diesen Befehl ausführen, müsste auf einem System, das die Compliance-Anforderungen erfüllt, folgende Ausgabe erscheinen:

```
"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.
```

Bei einem fehlgeschlagenen Audit würde die folgende Ausgabe zurückgegeben:

```
"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
```

Im obigen Fall wiesen die beiden Strings „foobar“ und „public“ nicht ausreichend zufällige Tokens auf, weswegen der Test nicht erfolgreich war.

Beispiel 4: SSH-Zugriffssteuerung mit Kontexttest überprüfen

Die folgende einfache **`.audit`**-Datei sucht unter Verwendung des Schlüsselworts **`context`** nach allen Konfigurationselementen des Typs „line“ und führt einen regulären Ausdruck (**`regex`**) aus, um festzustellen, ob die SSH-Zugriffssteuerung festgelegt ist.

```
<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>
```

Wenn Sie diesen Befehl ausführen, müsste auf einem System, das die Compliance-Anforderungen erfüllt, folgende Ausgabe erscheinen:

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

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

Bei einem fehlgeschlagenen Audit würde die folgende Ausgabe zurückgegeben:

```
"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
```

Im obigen Fall gab es zwei Strings, die dem regulären Ausdruck „**`line .*`**“ für das Schlüsselwort „**`context`**“ entsprachen. Da jedoch keiner davon dem regulären Ausdruck „**`item`**“ entsprach, wurde vom Audit die Meldung „FAILED“ zurückgegeben.

Bedingungen

Sie können eine **`if`**/**`then`**/**`else`**-Logik in der Cisco-Auditrichtlinie definieren. Dies ermöglicht es, dem Endbenutzer statt des Ergebnisses („Passed“/„Failed“) eine Warnung anzuzeigen, falls das Audit erfolgreich ist.

Die Syntax zur Ausführung von Bedingungen lautet wie folgt:

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

Beispiel:

```
<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>
```

Ob die Bedingung erfüllt wird oder nicht, wird im Bericht nicht angezeigt, da es sich hier um einen im Hintergrund stattfindenden Test handelt.

Bei den Bedingungen wird zwischen den Typen „**and**“ und „**or**“ unterschieden.

Referenz zu Audit-Compliancedateien für die Juniper-Konfiguration

In diesem Abschnitt werden Format und Funktionen von Juniper-Compliancetests und das Prinzip hinter den einzelnen Einstellungen erläutert.

Verwendung von Anführungszeichen:

Um Auditfelder gesetzte einzelne bzw. doppelte Anführungszeichen werden austauschbar verwendet. Ausgenommen sind lediglich die folgenden Fälle:

1. Bei Windows-Compliancetests, in denen Sonderfelder (z. B. CRLF) literal interpretiert werden müssen, müssen einzelne Anführungszeichen gesetzt werden. Eingebettete Felder, die als Strings zu interpretieren sind, müssen mit Escapezeichen versehen werden.

Beispiel:

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

2. Doppelte Anführungszeichen sind erforderlich, wenn „include_paths“ und „exclude_paths“ für WindowsFiles verwendet werden.

Wenn Strings in einem beliebigen Feldtyp (`description`, `value_data`, `regex` usw.) verwendet werden, die doppelte Anführungszeichen aufweisen, kann auf zweierlei Weise verfahren werden:

a. Verwenden Sie den jeweils anderen Anführungszeichentyp als äußere umschließende Anführungszeichen.

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

b. Stellen Sie eingebetteten doppelten Anführungszeichen den umgekehrten Schrägstrich („\“) als Escapezeichen voran.

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

Testtyp: CONFIG_CHECK

Juniper-Compliancetests sind in **`custom_item`**-Elementen gekapselt und haben den Typ CONFIG_CHECK oder SHOW_CONFIG_CHECK . Sie werden wie alle anderen **`.audit`**-Dateien verarbeitet und funktionieren auf Systemen mit dem Juniper-Betriebssystem (Junos). Der „CONFIG_CHECK“-Test umfasst mindestens zwei Schlüsselwörter. Die Schlüsselwörter **`type`** und **`description`** sind Pflichtangaben, auf die mindestens ein weiteres Schlüsselwort folgt. Der Test wird durch Prüfung der Konfiguration im „set“-Format ausgeführt.

Die Konfiguration kann im „set“-Format abgerufen werden, indem Sie „display set“ an die Anfrage „show configuration“ anhängen. Beispiel:

```
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"
```

Schlüsselwörter

Die folgende Tabelle gibt an, wie die einzelnen Schlüsselwörter in Juniper-Compliancetests verwendet werden können:

Schlüsselwort	Einsatzbeispiel und unterstützte Einstellungen
`type`	CHECK_CONFIG und SHOW_CHECK_CONFIG CHECK_CONFIG bestimmt, ob das angegebene Konfigurationselement in der Juniper-Ausgabe von „show configuration“ im „set“-Format vorhanden ist. SHOW_CONFIG_CHECK prüft auf gleiche Weise, ob das Konfigurationselement in der Ausgabe von „show configuration“ im Standardformat vorhanden ist.
`description`	Das Schlüsselwort „**`description`**“ bietet die Möglichkeit, dem ausgeführten Test eine kurze Beschreibung hinzuzufügen. Es wird dringend empfohlen, dem Feld **`description`** einen eindeutigen Wert zuzuweisen und sicherzustellen, dass andere Tests keinen identischen Eintrag im Feld **`description`** aufweisen. Auf der Basis des

	Feldes **description** generiert Tenable SecurityCenter automatisch eine eindeutige Plugin-ID. Beispiel: `description: " 3.1 Disable Unused Interfaces"`

info	Das Schlüsselwort „**info**“ wird verwendet, um dem ausgeführten Test eine ausführlichere Beschreibung hinzuzufügen. Dies könnte ein Verweis auf eine Rechtsvorschrift, eine URL mit weiteren Informationen, eine Unternehmensrichtlinie usw. sein. Mehrere **info**-Felder lassen sich in separaten Zeilen hinzufügen, um den Text als Abschnitt zu formatieren. Die Anzahl der verwendbaren **info**-Felder ist nicht begrenzt.
	
	 
	Jedes „**info**“-Tag muss in eine einzelne Zeile ohne Zeilenumbruch geschrieben werden. Wenn mehrere Zeilen erforderlich sind (z. B. zu Formatierungszwecken), fügen Sie einfach weitere „**info**“-Tags hinzu.
	
	Beispiel:
	`info: "Review the the list of interfaces"`
	`info: "Disable unused interfaces"`

severity	Das Schlüsselwort „**severity**“ gibt die Intensität (Schweregrad) des ausgeführten Tests an. Beispiel: `severity: MEDIUM` Die Intensität kann auf HIGH („Hoch“), MEDIUM („Moderat“) oder LOW („Niedrig“) festgelegt werden.
regex	Mit dem Schlüsselwort „**regex**“ aktivieren Sie die Suche nach der Konfigurationselementeinstellung, die einem bestimmten regulären Ausdruck entspricht. Beispiel: `regex: " set system syslog .+"` Die folgenden Metazeichen erfordern einen gesonderten Umgang: + \ * ( ) ^ Wenn diese Zeichen literal ausgewertet werden sollen, müssen Sie sie entweder mit zwei umgekehrten Schrägstrichen („\\“) als Escapezeichen auszeichnen oder sie in eckige Klammern („[]“) setzen. Die folgenden Zeichen hingegen müssen für eine literale Auswertung mit nur einem umgekehrten Schrägstrich ausgezeichnet werden: . ? " ' . Dies hat mit der Art und Weise zu tun, wie der Compiler diese Zeichen auswertet. Wenn das Tag „**regex**“ für einen Test festgelegt, aber keines der Tags „**expect**“, „**not_expect**“ oder „**number_of_lines**“ angegeben wurde, meldet der Test einfach alle Zeilen mit diesem regulären Ausdruck.
expect	Das Schlüsselwort ermöglicht die Prüfung des Konfigurationselements mit dem passenden „**regex**“-Tag. Wurde kein „**regex**“-Tag verwendet, sucht der Test in der gesamten Konfiguration nach dem „**expect**“-String.

	Beispiel:
	`expect: "syslog host 1.1.1.1"`
	
	Der Test ist bestanden, sofern die von „**regex**“ gefundene Konfigurationszeile dem Tag „**expect**“ entspricht oder, wenn „**regex**“ nicht verwendet wurde, der „**expect**“-String in der Konfiguration gefunden wurde.
	
	Beispiel:
	`regex: "syslog host [0-9\.]+"`
	`expect: "syslog host 1.1.1.1"`
	
	Im oben stehenden Beispiel stellt das Tag „**expect**“ sicher, dass der syslog-Host auf 1.1.1.1 festgelegt ist.

not_expect	Dieses Schlüsselwort ermöglicht die Suche nach Konfigurationselementen, die nicht in der Konfiguration vorhanden sein sollten.
	
	Beispiel:
	`not_expect: "syslog host 1.1.1.1"`
	
	Es bewirkt das Gegenteil von „**expect**“. Der Test ist bestanden, sofern die von „**regex**“ gefundene Konfigurationszeile dem Tag „**not expect**“ nicht entspricht oder, wenn „**regex**“ nicht verwendet wurde, der „**not expect**“-String in der Konfiguration nicht gefunden wurde.
	
	Beispiel:
	`regex: "syslog host [0-9\.]+"`
	`not_expect: "syslog host 1.1.1.1"`
	
	Im oben stehenden Beispiel stellt das Tag „**not_expect**“ sicher, dass der syslog-Host nicht auf 1.1.1.1 festgelegt ist.

number_of_lines	Dieses Schlüsselwort ermöglicht die Überprüfung der Compliance eines Audittests nach Anzahl der übereinstimmenden Zeilen, die von der Konfiguration zurückgegeben wurden.
	
	`<custom_item>`
	` type: CONFIG_CHECK`
	` description: "Syslog"`
	` regex: "syslog host [0-9\.]+"`
	` number_of_lines: "^1$"`
	`</custom_item>`
	
	Im oben stehenden Beispiel ist der Test erfolgreich, sofern genau eine Zeile zurückgegeben wird, die mit „**regex**“ übereinstimmt.

CONFIG_CHECK Beispiele

Folgende Beispiele zeigen die Verwendung von CONFIG_CHECK auf einem Juniper-Gerät:

```
<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>
```

Testtyp: SHOW_CONFIG_CHECK

Dieser Test prüft weitgehend dieselben Einstellungen wie der Audittest CONFIG_CHECK. Das Format der geprüften Konfiguration unterscheidet sich jedoch. SHOW_CONFIG_CHECK prüft die Konfiguration in ihrem Standardformat.

So sieht die Konfiguration beispielsweise wie folgt im Standardformat aus:

```
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;
}
```

Dieser Test wird nur empfohlen, wenn Sie verglichen mit CONFIG_CHECK mehr Flexibilität benötigen. Da jeder einzelne SHOW_CONFIG_CHECK-Audittest die Ausführung eines separaten Befehls auf dem Juniper-Gerät zur Folge hat, führt der Prozess möglicherweise zu einer höheren CPU-Belastung und dauert eventuell länger. Mit diesem Test erhält der Auditor mehr Flexibilität. Zudem werden potenzielle zukünftige Fälle unterstützt, die via CONFIG_CHECK unter Umständen nicht effizient überprüft werden können.

Schlüsselwörter

Die folgende Tabelle gibt an, wie die einzelnen Schlüsselwörter in Junos-Compliancetests verwendet werden können. Beachten Sie, dass die Compliance eines Tests durch Vergleich der Testausgabe mit einem der Tags „**expect**“, „**not_expect**“ oder „**number_of_lines**“ bestimmt werden kann. Es kann maximal ein Tag für den Compliancetest verwendet werden (d. h., es sind wahlweise „**expect**“, „**not_expect**“ oder „**number_of_lines**“ möglich, nicht aber „**expect**“ und „**not_expect**“).

Schlüsselwort	Einsatzbeispiel und unterstützte Einstellungen

hierarchy	Dieses Schlüsselwort ermöglicht es Benutzern, zu einer bestimmten Hierarchie in der Junos-Konfiguration zu navigieren.
	
	Beispiel:
	`hierarchy: "interfaces"`
	
	Das Schlüsselwort „hierarchy“ wird in einem SHOW_CONFIG_CHECK intern an den Befehl „show configuration“ angehängt. Beispiel:
	
	`<custom_item>`
	` type: SHOW_CONFIG_CHECK`
	` description: "3.6 Forbid Multiple Loopback Addresses"`
	` hierarchy: "interfaces"`
	`</custom_item>`
	
	Der oben stehende Test entspricht der Ausführung des folgenden Befehls:
	
	`show configuration interfaces`

property	Dieses Schlüsselwort ermöglicht es Benutzern, eine spezifische Eigenschaft („**property**“) des Junos-Geräts zu prüfen. SHOW_CONFIG_CHECK prüft automatisch den „show configuration“-Befehl gefolgt von mindestens einem Schlüsselwort wie **match**, **except** und **find**. Sofern das Schlüsselwort „**property**“ festgelegt wurde, prüft der Test auf die spezifische Eigenschaft.
	
	Beispiel:
	`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>`
	
	Der oben stehende Test entspricht der Ausführung des folgenden Befehls:
	
	`show ospf interface detail`

	Beachten Sie, dass im oben stehenden Beispiel „show configuration“ im Gegensatz zu den anderen Beispielen nicht ausgeführt wurde.

find	Dieses Schlüsselwort sucht nach der passenden Konfigurationshierarchie in einem SHOW_CONFIG_CHECK-Audittest.
	`find: "chap"`
	Das Schlüsselwort **find** wird an die „show configuration“-Anfrage angehängt.
	`<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>`
	Der oben stehende Test entspricht der Ausführung des folgenden Befehls:
	`show configuration interfaces | find "chap" | match "access-profile"`

match	Dieses Schlüsselwort sucht nach übereinstimmenden Zeilen in einem SHOW_CONFIG_CHECK-Audittest.
	`match: "multihop"`
	Das Schlüsselwort **match** wird an die „show configuration“-Anfrage angehängt.
	`<custom_item>`
	` type: SHOW_CONFIG_CHECK`
	` description: "3.6 Forbid Multiple Loopback Addresses"`
	` hierarchy: "interfaces"`
	` match: "lo[0-9]"`
	`</custom_item>`
	Der oben stehende Test entspricht der Ausführung des folgenden Befehls:
	`show configuration interfaces | match "lo[0-9]"`

except	Dieses Schlüsselwort schließt bestimmte Zeilen aus der Konfiguration in einem SHOW_CONFIG_CHECK-Audittest aus. `except: "multihop"` Das **except** Schlüsselwort wird an die „show configuration“ Anfrage angehängt.

	<custom_item> type: SHOW_CONFIG_CHECK description: "6.8.1 Require External Time Sources" hierarchy: "system ntp" match: "server" except: "boot-server" </custom_item> Der oben stehende Test entspricht der Ausführung des folgenden Befehls: show configuration system ntp | match "server" | except "boot-server"
expect	Das Schlüsselwort ermöglicht die Prüfung des Konfigurationselements mit dem passenden „**regex**“-Tag. Wurde kein „**regex**“-Tag verwendet , sucht der Test in der gesamten Konfiguration nach dem „**expect**“-String. Der Test ist bestanden, sofern die von „**regex**“ gefundene Konfigurationszeile dem Tag „**expect**“ entspricht oder, wenn „**regex**“ nicht verwendet wurde, der „**expect**“-String in der Konfiguration gefunden wurde. regex: "syslog host [0-9\.]+" expect: "syslog host 1.2.4.5" Im oben stehenden Beispiel stellt das Tag „**expect**“ sicher, dass die Komplexität auf einen Wert zwischen 1 und 4 gesetzt ist. expect: "syslog host" Im oben stehenden Beispiel stellt das Tag „**expect**“ sicher, dass die Komplexität auf 4 gesetzt ist.
not_expect	Dieses Schlüsselwort ermöglicht die Suche nach Konfigurationselementen, die nicht in der Konfiguration vorhanden sein sollten. Es bewirkt das Gegenteil von „**expect**“. Der Test ist bestanden, sofern die von „**regex**“ gefundene Konfigurationszeile dem Tag „**not expect**“ nicht entspricht oder, wenn „**regex**“ nicht verwendet wurde, der „**not expect**“-String in der Konfiguration nicht gefunden wurde. regex: "syslog host [0-9\.]+" not_expect: "syslog host 1.2.3.4" not_expect: "syslog host"
number_of_lines	Dieses Schlüsselwort ermöglicht den Compliancetest eines Audittests je nach Anzahl der übereinstimmenden Zeilen, die von der Konfiguration zurückgegeben wurden. <custom_item>

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

Im oben stehenden Beispiel ist der Test erfolgreich, sofern eine mit „**regex**“ übereinstimmende Zeile zurückgegeben wird.

SHOW_CONFIG_CHECK Beispiele

Folgende Beispiele zeigen die Verwendung von SHOW_CONFIG_CHECK auf einem Juniper-Gerät:

```
<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>
```

Bedingungen

Sie können eine **if/then/else**-Logik in der Juniper-Auditrichtlinie definieren. Auf diese Weise muss der Endbenutzer nur eine einzige Datei verwenden, die mehrere Konfigurationen verarbeiten kann.

Die Syntax zur Ausführung von Bedingungen lautet wie folgt:

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

Beispiel:

```
<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>
```

Die Bedingung taucht nie im Bericht auf, d. h., es handelt sich um einen im Hintergrund stattfindenden Test, bei dem Erfolg oder Fehlschlagen nicht gemeldet werden.

Bei den Bedingungen wird zwischen den Typen „**and**“ und „**or**“ unterschieden.

Berichterstellung

Kann in einem <then>- oder <else>-Abschnitt durchgeführt werden, um die gewünschte PASSED/FAILED-Bedingung zu erhalten.

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

PASSED, WARNING und FAILED sind zulässige Werte für „report type“.

Referenz zu .audit-Compliancedateien für die Check Point GAiA-Konfiguration

In diesem Abschnitt werden Format und Funktionen von Check Point GAiA-Compliancetests und das Prinzip hinter den einzelnen Einstellungen erläutert.

Verwendung von Anführungszeichen:

Um Auditfelder gesetzte einzelne bzw. doppelte Anführungszeichen werden austauschbar verwendet. Ausgenommen sind lediglich die folgenden Fälle:

1. Bei Windows-Compliancetests, in denen Sonderfelder (z. B. CRLF) literal interpretiert werden müssen, müssen einzelne Anführungszeichen gesetzt werden. Eingebettete Felder, die als Strings zu interpretieren sind, müssen mit Escapezeichen versehen werden.

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

2. Doppelte Anführungszeichen sind erforderlich, wenn „include_paths“ und „exclude_paths“ für WindowsFiles verwendet werden.

Wenn Strings in einem beliebigen Feldtyp (`description`, `value_data`, `regex` usw.) verwendet werden, die doppelte Anführungszeichen aufweisen, kann auf zweierlei Weise verfahren werden:

a. Verwenden Sie den jeweils anderen Anführungszeichentyp als äußere umschließende Anführungszeichen.

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

b. Stellen Sie eingebetteten doppelten Anführungszeichen den umgekehrten Schrägstrich („\“) als Escapezeichen voran.

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

Testtyp: CONFIG_CHECK

Check Point-Compliancetests sind in **custom_item**-Elementen gekapselt und haben den Typ CONFIG_CHECK. Sie werden wie alle anderen **.audit** Dateien verarbeitet und funktionieren auf Systemen mit dem Check Point GAiA-Betriebssystem. Der „CONFIG_CHECK“-Test umfasst mindestens zwei Schlüsselwörter. Die Schlüsselwörter **type** und **description** sind Pflichtangaben, auf die mindestens ein weiteres Schlüsselwort folgt. Der Test prüft die Ausgabe des Befehls „**show config**“, die sich standardmäßig im Format „set“ befindet.

Schlüsselwörter

Die folgende Tabelle gibt an, wie die einzelnen Schlüsselwörter in GAiA-Compliancetests verwendet werden können:

Schlüsselwort	Einsatzbeispiel und unterstützte Einstellungen
type	„CHECK_CONFIG“ bestimmt, ob das angegebene Konfigurationselement in der GAiA-Ausgabe von „show configuration“ vorhanden ist.
description	Das Schlüsselwort „**description**“ bietet die Möglichkeit, dem ausgeführten Test eine kurze Beschreibung hinzuzufügen. Es wird dringend empfohlen, dem Feld **description** einen eindeutigen Wert zuzuweisen und sicherzustellen, dass andere Tests keinen identischen Eintrag im Feld **description** aufweisen. Auf der Basis des Feldes **description** generiert Tenable SecurityCenter automatisch eine eindeutige Plugin-ID. Beispiel: `description: "1.0 Require strong Password Controls - 'min-password-length >= 8'"`
info	Das Schlüsselwort „**info**“ wird verwendet, um dem ausgeführten Test eine ausführlichere Beschreibung hinzuzufügen. Dies könnte ein Verweis auf eine Rechtsvorschrift, eine URL mit weiteren Informationen, eine Unternehmensrichtlinie usw. sein. Mehrere **info**-Felder lassen sich in separaten Zeilen hinzufügen, um den Text als Abschnitt zu formatieren. Die Anzahl der verwendbaren **info**-Felder ist nicht begrenzt.  Jedes „**info**“-Tag muss in eine einzelne Zeile ohne Zeilenumbruch geschrieben werden. Wenn mehrere Zeilen erforderlich sind (z. B. zu Formatierungszwecken), fügen Sie einfach weitere „**info**“-Tags hinzu. Beispiel: `info: "Enable palindrome-check on passwords"`
severity	Das Schlüsselwort „**severity**“ gibt die Intensität (Schweregrad) des ausgeführten Tests an. Beispiel: `severity: MEDIUM` Die Intensität kann auf HIGH („Hoch“), MEDIUM („Moderat“) oder LOW („Niedrig“) festgelegt werden.

regex	Mit dem Schlüsselwort „**regex**“ aktivieren Sie die Suche nach der Konfigurationselementeinstellung, die einem bestimmten regulären Ausdruck entspricht. Beispiel: `regex: "set snmp .+"` Die folgenden Metazeichen erfordern einen gesonderten Umgang: + \ * ( ) ^ Wenn diese Zeichen literal ausgewertet werden sollen, müssen Sie sie entweder mit zwei umgekehrten Schrägstrichen („\\“) als Escapezeichen auszeichnen oder sie in eckige Klammern („[]“) setzen. Die folgenden Zeichen hingegen müssen für eine literale Auswertung mit nur einem umgekehrten Schrägstrich ausgezeichnet werden: . ? " ' . Dies hat mit der Art und Weise zu tun, wie der Compiler diese Zeichen auswertet. Wenn das Tag „`regex`“ für einen Test festgelegt, aber keines der Tags „expect“, „not_expect“ oder „number_of_lines“ angegeben wurde, meldet der Test einfach alle Zeilen mit diesem regulären Ausdruck.

expect	Das Schlüsselwort ermöglicht die Prüfung des Konfigurationselements mit dem passenden „**regex**“-Tag. Wurde kein „**regex**“-Tag verwendet, sucht der Test in der gesamten Konfiguration nach dem „**expect**“-String.
	
	Der Test ist bestanden, sofern die von „**regex**“ gefundene Konfigurationszeile dem Tag „**expect**“ entspricht oder, wenn „**regex**“ nicht verwendet wurde, der „**expect**“-String in der Konfiguration gefunden wurde.
	
	Beispiel:
	`regex: "set password-controls complexity"`
	`expect: "set password-controls complexity [1-4]"`
	
	Im oben stehenden Beispiel stellt das Tag „**expect**“ sicher, dass die Komplexität auf einen Wert zwischen 1 und 4 festgelegt ist.

not_expect	Dieses Schlüsselwort ermöglicht die Suche nach Konfigurationselementen, die nicht in der Konfiguration vorhanden sein sollten.
	
	Es bewirkt das Gegenteil von „**expect**“. Der Test ist bestanden, sofern die von „**regex**“ gefundene Konfigurationszeile dem Tag „**not expect**“ nicht entspricht oder, wenn „**regex**“ nicht verwendet wurde, der „**not expect**“-String in der Konfiguration nicht gefunden wurde.
	
	Beispiel:
	`regex: "set password-controls password-expiration"`
	`not_expect: "set password-controls password-expiration never"`
	
	Im oben stehenden Beispiel stellt das Tag „**not_expect**“ sicher, dass „password-controls“ nicht auf „never“ festgelegt ist.

CONFIG_CHECK Beispiele

Folgende Beispiele zeigen die Verwendung von CONFIG_CHECK auf einem Check Point-Gerät:

```
<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>
```

Bedingungen

Sie können eine **`if`**/**`then`**/**`else`**-Logik in der Check Point-Auditrichtlinie definieren. Auf diese Weise muss der Endbenutzer nur eine einzige Datei verwenden, die mehrere Konfigurationen verarbeiten kann.

Die Syntax zur Ausführung von Bedingungen lautet wie folgt:

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

Beispiel:

```
<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>
```

Die Bedingung taucht nie im Bericht auf, d. h. es ist ein im Hintergrund stattfindender Test, bei dem Erfolg oder Fehlschlagen nicht gemeldet werden.

Bei den Bedingungen wird zwischen den Typen „**and**“ und „**or**“ unterschieden.

Berichterstellung

Kann in einem <then>- oder <else>-Abschnitt durchgeführt werden, um die gewünschte PASSED/FAILED-Bedingung zu erhalten.

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

PASSED, WARNING und FAILED sind zulässige Werte für „report type“.

Referenz zu Compliancedateien für Palo Alto Firewall-Konfigurationsaudits

Die Compliancetests für Palo Alto unterscheiden sich von anderen Compliance-Audits. Einer der wesentlichen Unterschiede besteht in der umfassenden Nutzung von XSLTs (XSL-Transformationen) zum Extrahieren wichtiger Daten (weitere Informationen finden Sie in Anhang C). Die Palo Alto Firewall beantwortet API-Anfragen in den meisten Fällen im XML-Format, wodurch XSLTs zur sinnvollsten Auditmethode werden. Sofern Sie mit XSLTs nicht vertraut sind, können Sie es als Abfrage einer XML-Datei zum Extrahieren der gewünschten Daten im gewünschten Dateiformat betrachten. Kurz gesagt ist XSLT das, was SQL für Datenbanken darstellt.

Das Palo Alto-Audit unterstützt zwei Arten von Tests: AUDIT_XML und AUDIT_REPORTS.

AUDIT_XML

Es folgt ein Beispiel für einen Palo Alto AUDIT_XML-Test:

```
<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>
```

Das Audit setzt sich im Wesentlichen aus vier Teilen zusammen:

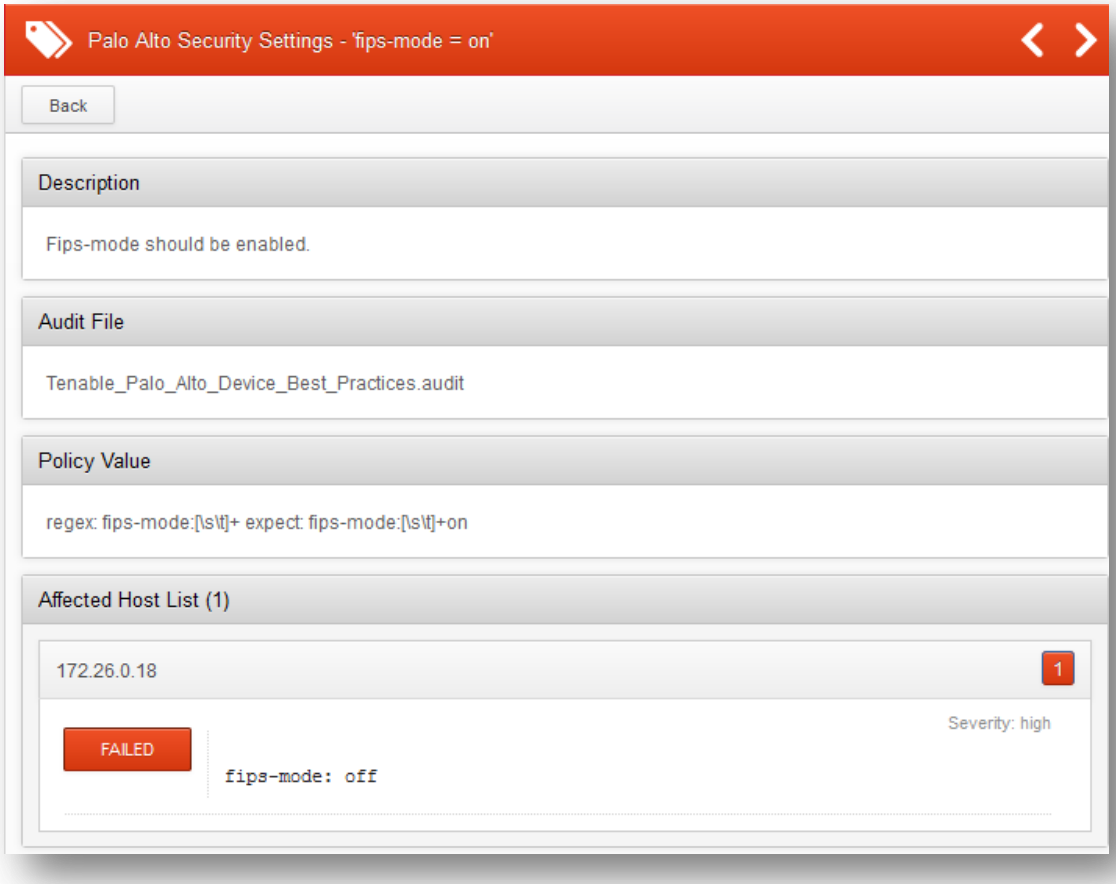
1. **`type`** beschreibt den Audittyp (in diesem Fall wird XML geprüft) und enthält eine Beschreibung (**`description`**) des Audits. Über das Schlüsselwort **`info`** können Sie relevanten Text in den Bericht aufnehmen.

2. **`api_request_type`** beschreibt die Art der Anfrage (op == operational config). Dies ist die eigentliche Anfrage, die am Ende ausgeführt wird. Es handelt sich um die einzige Anfrage, die gegenwärtig unterstützt wird.

3. Das Schlüsselwort **`xsl_stmt`** ermöglicht eine Definition der XSL-Transformation, das auf die XML-Daten angewendet wird, die nach Ausführung der API-Anfrage zurückgegeben werden.

4. Die Schlüsselwörter **`regex`** und **`expect`** schließlich gestatten die Durchführung von Compliance- und Konfigurationstests.

Der obige Beispieltest ergibt folgenden Bericht in Nessus:



AUDIT_REPORTS

Eine der praktischen Funktionen der Palo Alto-Firewall ist die fortlaufende Erstellung von Netzwerkprofilen, aufgrund derer mehr als 40 vordefinierte Berichte pro Tag generiert werden. Zu den Berichten zählen „Top Applications“ („Wichtigste Anwendungen“), „Top Attackers“ („Häufigste Angreifer“) und „Spyware Infected Hosts“ („Mit Spyware infizierte Hosts“). Administratoren können auf Wunsch auch dynamische Berichte generieren (z. B. Berichte zur vergangenen Stunde). Nessus kann diese Berichte jetzt direkt abrufen und sie in den Nessus-Bericht aufnehmen.

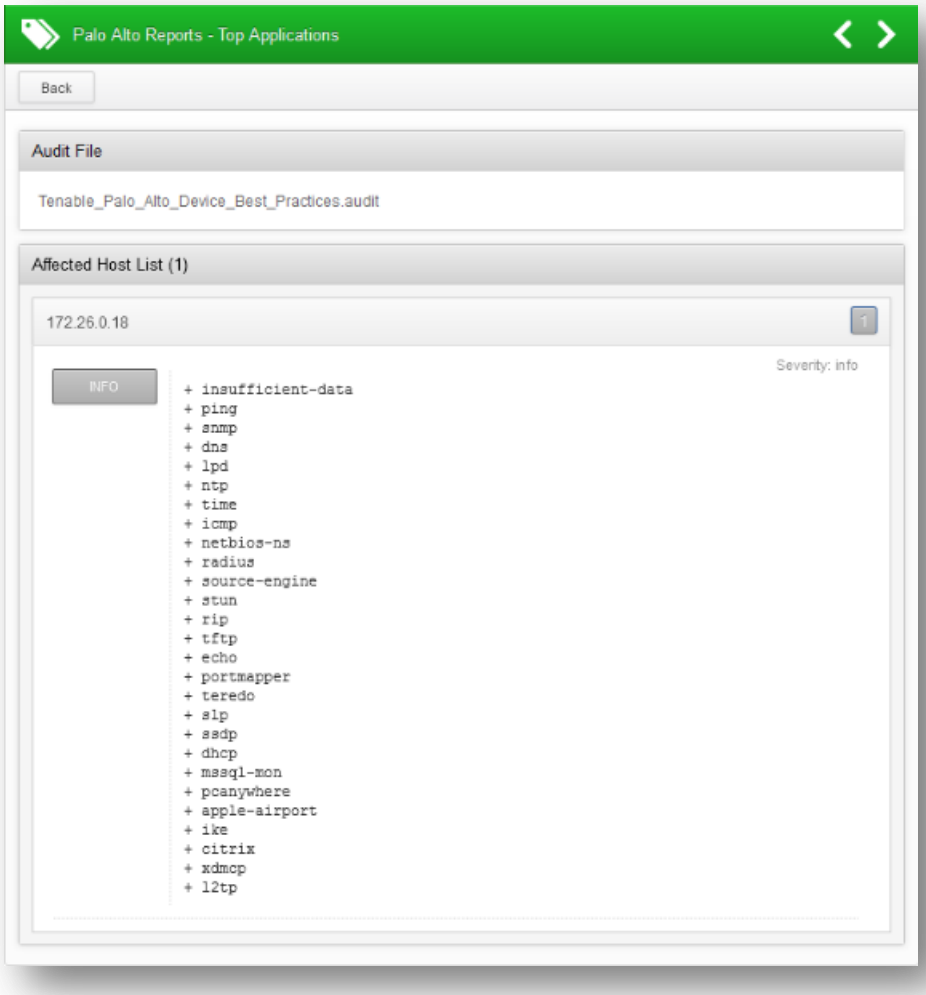
Die Funktion bietet zwei Vorteile. Zum einen brauchen Benutzer nicht zwischen zwei Oberflächen zu wechseln, um auf dieselben Daten zuzugreifen, zum anderen haben sie so Gelegenheit, den Bericht per Audit zu prüfen. Wenn beispielsweise Facebook als Anwendung in einem Netzwerk nicht genutzt werden darf, können Administratoren festlegen, dass ein Test nicht bestanden ist, wenn Facebook im Bericht „Top Applications“ auftaucht. Beispiel:

```
<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>
```

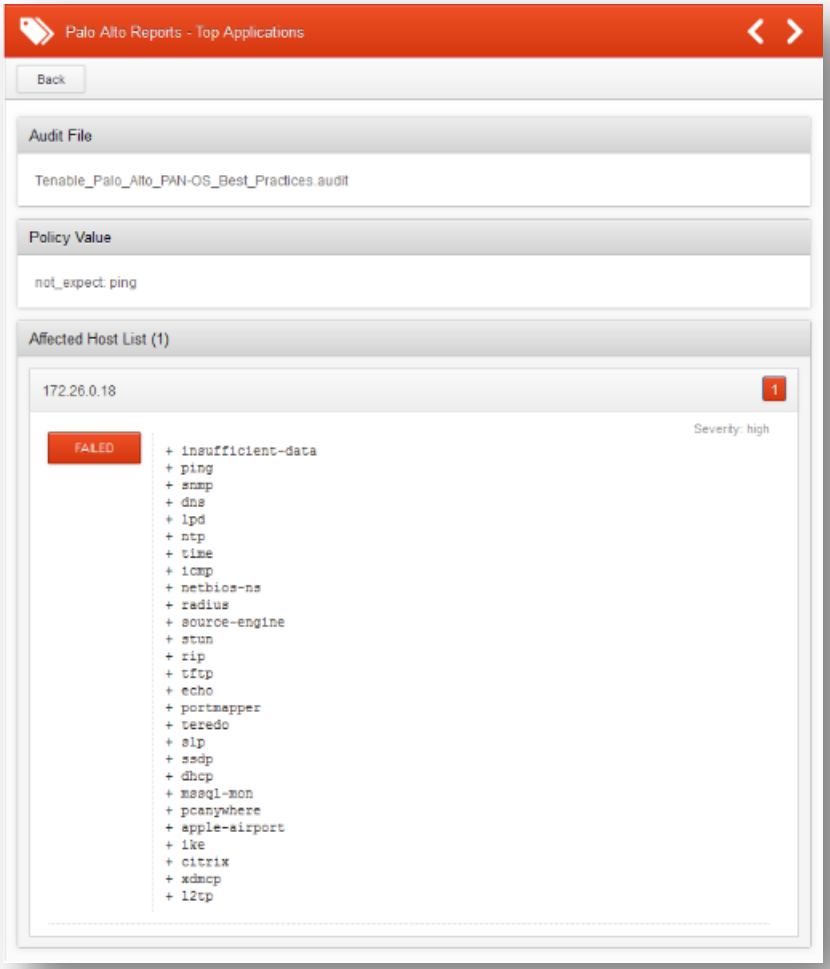
Der Bericht kann für die Nutzung des Schlüsselworts **not_expect** modifiziert werden:

```
<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>
```

Im ersten Beispiel wird ein Bericht wie der folgende zurückgegeben:



Das zweite Beispiel ergibt einen Bericht mit Angaben zum nicht bestandenen Test:



Schlüsselwörter

Folgende Schlüsselwörter werden in Palo Alto-Audits unterstützt:

Schlüsselwort	Beschreibung
type	Muss immer auf AUDIT_XML oder AUDIT_REPORTS festgelegt sein.
description	Diese Angabe wird als Titel für eindeutige Compliancesicherheitslücken in SecurityCenter verwendet. Außerdem ist dies der erste von Nessus gemeldete Datensatz.
info	Das Schlüsselwort ermöglicht es Benutzern, dem ausgeführten Test eine ausführlichere Beschreibung hinzuzufügen. Es sind mehrere **info**-Felder ohne vordefinierte Einschränkung zulässig. Der Inhalt eines **info**-Feldes sollte in doppelte Anführungszeichen gesetzt werden.

api_request_type	Dieses Schlüsselwort beschreibt den Typ der Anfrage. Die Palo Alto-API unterstützt sechs Anfragetypen: `keygen`, `op`, `commit`, `reports`, `export` und `config`. Für den Zweck dieses Plugins wird nur der Anfragetyp **op** verwendet.
request	Dieses Schlüsselwort nennt die Anfrage zur Ausführung auf der Firewall. Das Ergebnis der einzelnen Anfragen wird in einem Cache zwischengespeichert, damit anschließende Anfragen nicht erneut eine Anfrage ergeben. Zudem ist das standardmäßige Audit von Tenable im Falle des AUDIT_REPORTS-Tests auf neun Tests beschränkt. Um mehr Berichte aufzunehmen, werden Benutzer aufgefordert, neue Tests zu erstellen und das Schlüsselwort `request` m Anschluss an `type=report` durch die REST API-URL zu ersetzen. Beispiel: `/api/?type=report&reporttype=predefined&reportname=hruser-top-url-categories`
regex	Mit diesem Schlüsselwort aktivieren Sie die Suche nach Elementen, die einem bestimmten regulären Ausdruck entsprechen. Wenn das Schlüsselwort „**regex**“ für einen Test festgelegt, aber keines der Tags „**expect**“ oder „**not_expect**“ angegeben wurde, meldet der Test alle Zeilen mit diesem regulären Ausdruck.

Beachten Sie, dass die Compliance eines Tests durch Vergleich der Testausgabe mit einem der Tags „**expect**“ oder „**not_expect**“ bestimmt werden kann. Es kann maximal ein Tag für den Compliancetest verwendet werden (d. h. es sind wahlweise „**expect**“ **oder** „**not_expect**“ möglich, nicht aber „**expect**“ **und** „**not_expect**“).

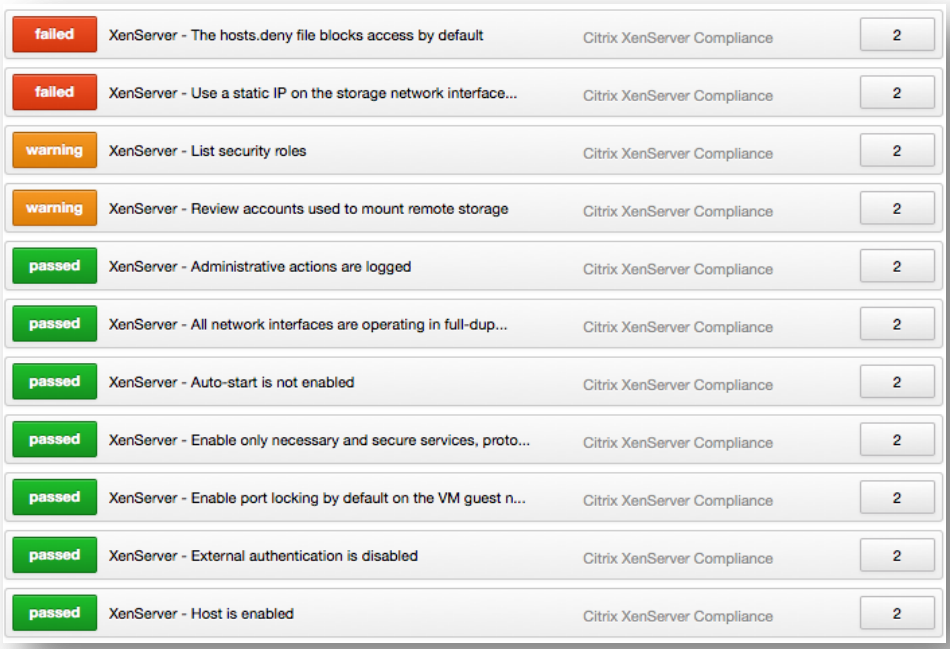
Schlüsselwort	Beschreibung
expect	Das Schlüsselwort ermöglicht die Prüfung des Konfigurationselements mit dem passenden „**regex**“-Schlüsselwort. Wurde kein „**regex**“-Schlüsselwort verwendet , sucht der Test in der gesamten Konfiguration nach dem „**expect**“-String. Der Test ist bestanden, sofern die von „**regex**“ gefundene Konfigurationszeile dem „**expect**“-String entspricht oder, wenn „**regex**“ nicht verwendet wurde, der „**expect**“-String in der Konfiguration gefunden wurde.
not_expect	Dieses Schlüsselwort ermöglicht die Suche nach Konfigurationselementen, die **nicht** in der Konfiguration vorhanden sein sollten. Es bewirkt das Gegenteil von „**expect**“. Der Test ist bestanden, sofern die von „**regex**“ gefundene Konfigurationszeile dem „**not expect**“-String nicht entspricht oder, wenn das Schlüsselwort „**regex**“ nicht verwendet wurde, der „**not expect**“-String in der Konfiguration nicht gefunden wurde.

Referenz zu Compliancedateien für Citrix XenServer-Audits

Abgesehen von einer Ausnahme beruhen die Compliancetests für Citrix XenServer weitgehend auf dem Abschnitt Referenz zu Compliancedateien für UNIX-Konfigurationsaudits weiter unten. Ein weiteres Audit namens AUDIT_XE ist zur Durchführung von Patch-Audits verfügbar. Folgende Testtypen stehen für XenServer-Audits bereit:

- FILE_CHECK_NOT
- PROCESS_CHECK
- FILE_CONTENT_CHECK
- FILE_CONTENT_CHECK_NOT
- CMD_EXEC
- GRAMMAR_CHECK

- RPM_CHECK
- CHKCONFIG
- XINETD_SVC
- AUDIT_XE



Testtyp: AUDIT_XE

Es folgt ein Beispiel eines XenServer AUDIT_XE-Tests:

```
<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>
```

Schlüsselwörter

Die folgende Tabelle gibt an, wie die einzelnen Schlüsselwörter in Citrix XenServer-Compliancetests verwendet werden können:

Schlüsselwort	Einsatzbeispiel und unterstützte Einstellungen
type	AUDIT_XE

description	Dieses Schlüsselwort bietet die Möglichkeit, dem ausgeführten Test eine kurze Beschreibung hinzuzufügen. Es wird empfohlen, dem Feld einen eindeutigen Wert zuzuweisen und sicherzustellen, dass andere Tests keinen identischen Eintrag in diesem Feld `description` aufweisen. Dies ist erforderlich, weil SecurityCenter auf der Basis des Feldes `description` automatisch eine Plugin-ID generiert. Beispiel: `description: "List running VMs"`
info	Das Schlüsselwort ermöglicht es Benutzern, dem ausgeführten Test eine ausführlichere Beschreibung hinzuzufügen. Es sind mehrere `info`-Felder ohne vordefinierte Einschränkung zulässig. Der Inhalt eines `info`-Feldes sollte in doppelte Anführungszeichen gesetzt werden. Beispiel: `info: "The allocated virtual CPUs (VCPU) should be reviewed. Desired settings depend on workload and operating system type."`
see_also	Dieses Schlüsselwort ermöglicht es Benutzern, Links aufzunehmen, die ggf. hilfreiche Angaben zu einem Test enthalten. Beispiel: `see_also: "http://support.citrix.com/article/CTX137828"`
reference	Dieses Schlüsselwort ermöglicht den Zusatz von Querverweisen zu Audittests. Beispiel: `reference: "PCI|2.2.3,SANS-CSC|1"`
solution	Das Schlüsselwort stellt Text zur Aufnahme von Textlösungen zur Compliance-Fehlerbehebung bereit.
severity	Mit diesem Schlüsselwort können Benutzer die Testintensität festlegen. Die Intensität kann auf HIGH („Hoch“), MEDIUM („Moderat“) oder LOW („Niedrig“) festgelegt werden. Beispiel: `severity: MEDIUM`
cmd	Dieses Schlüsselwort gibt den Befehl **xe** an, der auf dem Zielsystem ausgeführt wird. Beispiel: `cmd: "/usr/bin/xe subject-list params=all"`
regex	Mit diesem Schlüsselwort aktivieren Sie die Aufzählung von Elementen, die einem bestimmten regulären Ausdruck entsprechen. Wenn das Schlüsselwort „**regex**“ für einen Test festgelegt, aber keines der Tags „**expect**“ oder „**not_expect**“ angegeben wurde, meldet der Test alle diesem regulären Ausdruck entsprechenden Elemente. Beispiel: `regex: "power-state.+"`
expect	Wenn das Schlüsselwort `expect` angegeben wurde, wird der Test nur dann als bestanden gemeldet, wenn alle Ergebnisse mit dem Schlüsselwort „**expect**“ übereinstimmen. Wird keine Übereinstimmung mit **expect** erzielt, schlägt der Test für alle Ergebnisse fehl, die nicht mit **expect** übereinstimmen.

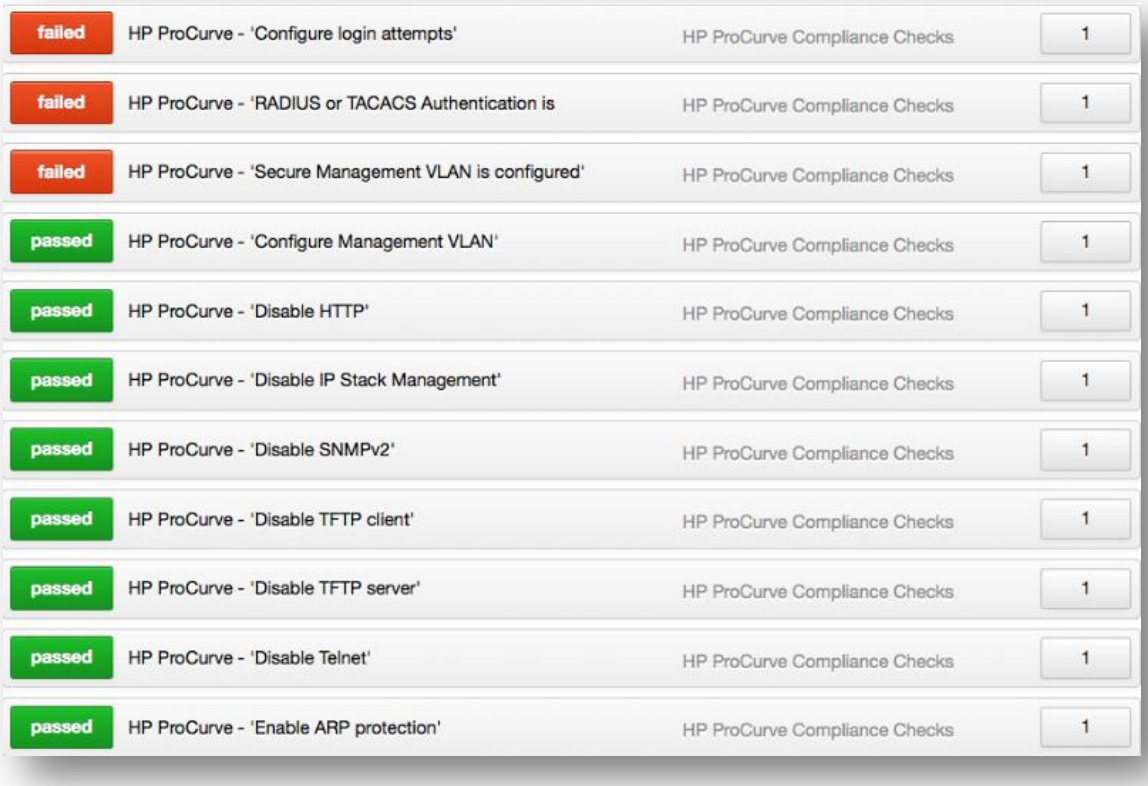
	Beispiel:
	
	`<custom_item>`
	` type: AUDIT_XE`
	` description: "List Running VMs - Any non running vms."`
	` cmd: "/usr/bin/xe vm-list params=uuid,name-label,is-a-`
	`  template,power-state,allowed-operations"`
	` regex: "power-state .+"`
	` expect: "running"`
	`</custom_item>`

not_expect	Wurde das Schlüsselwort `not_expect` angegeben, ist der Test bestanden, solange keines der Ergebnisse mit dem regulären Ausdruck `not_expect` übereinstimmt.
	
	Beispiel:
	
	`<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	Dieses Schlüsselwort ermöglicht es, bestimmte Elemente aus den Ergebnissen zu überspringen bzw. sie zu ignorieren.
	
	Beispiel:
	
	`<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>`

Referenz zu Compliancedateien für HP ProCurve-Audits

Das HP ProCurve-Audit ist in vielerlei Hinsicht eine Erweiterung des Cisco Compliance-Plugins. Die Tenable HP ProCurve-Auditdatei beruht auf einem HP-Whitepaper über das Härten ProCurve-Switches. Das Audit enthält Tests zur Deaktivierung von instabilen Diensten und Aktivierung der Zugriffssteuerung (z. B. TACACS, RADIUS). Gültige SSH-Anmeldedaten für einen „root“-Benutzer oder Administrator mit voller Zugriffsberechtigung werden benötigt.



Testtypen

HP ProCurve-Compliancetests verwenden einen von drei Testtypen. Standardmäßig wird die folgende Testsyntax verwendet:

```
<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>
```

Schlüsselwörter

Die folgende Tabelle gibt an, wie die einzelnen Schlüsselwörter in HP ProCurve-Compliancetests verwendet werden können:

Schlüsselwort	Einsatzbeispiel und unterstützte Einstellungen
type	CONFIG_CHECK CONFIG_CHECK_NOT RANDOMNESS_CHECK
description	Dieses Schlüsselwort bietet die Möglichkeit, dem ausgeführten Test eine kurze Beschreibung hinzuzufügen. Es wird empfohlen, dem Feld einen eindeutigen Wert

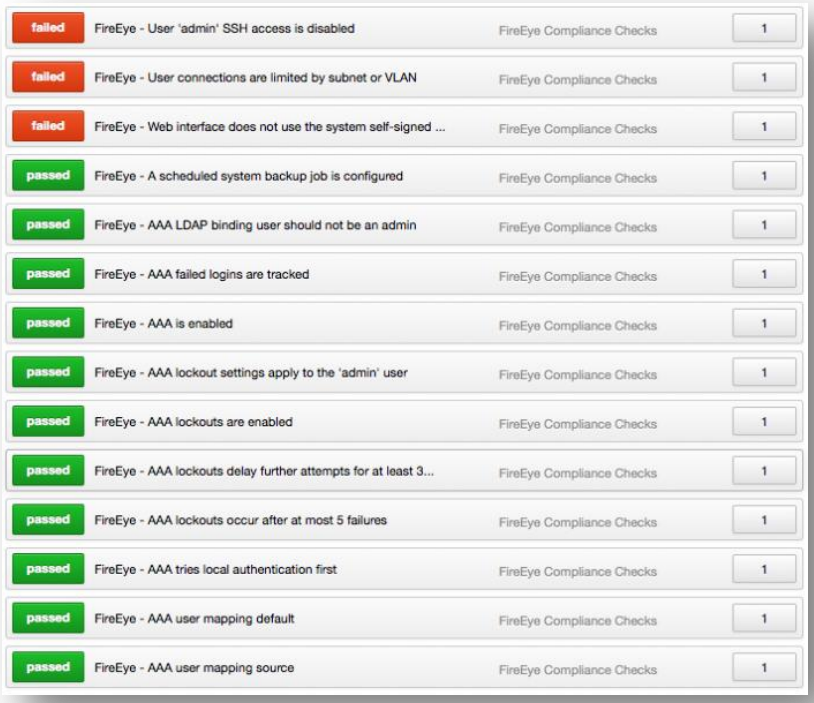
	zuzuweisen und sicherzustellen, dass andere Tests keinen identischen Eintrag in diesem Feld `description` aufweisen. Dies ist erforderlich, weil SecurityCenter auf der Basis des Feldes `description` automatisch eine Plugin-ID generiert. Beispiel: `description: " Verify login authentication"`
info	Das Schlüsselwort ermöglicht es Benutzern, dem ausgeführten Test eine ausführlichere Beschreibung hinzuzufügen. Es sind mehrere `info`-Felder ohne vordefinierte Einschränkung zulässig. Der Inhalt eines `info`-Feldes sollte in doppelte Anführungszeichen gesetzt werden. Beispiel: `info: "Verifies login authentication configuration."`
see_also	Dieses Schlüsselwort ermöglicht es Benutzern, Links aufzunehmen, die ggf. hilfreiche Angaben zu einem Test enthalten. Beispiel: `see_also: "http://www.hp.com/rnd/support/faqs/1800.htm"`
reference	Dieses Schlüsselwort ermöglicht den Zusatz von Querverweisen für Audittests. Beispiel: `reference: "PCI|2.2.3,SANS-CSC|1"`
solution	Das Schlüsselwort stellt Text zur Aufnahme von Textlösungen zur Compliance-Fehlerbehebung bereit. Beispiel: `solution: "Modify the configuration to add missing line"`
severity	Mit diesem Schlüsselwort können Benutzer die Testintensität festlegen. Die Intensität kann auf HIGH („Hoch“), MEDIUM („Moderat“) oder LOW („Niedrig“) festgelegt werden. Beispiel: `severity: MEDIUM`
regex	Mit diesem Schlüsselwort aktivieren Sie die Aufzählung von Elementen, die einem bestimmten regulären Ausdruck entsprechen. Wenn das Schlüsselwort „**`regex`**“ für einen Test festgelegt, aber keines der Tags „**`expect`**“ oder „**`not_expect`**“ angegeben wurde, meldet der Test alle diesem regulären Ausdruck entsprechenden Elemente. Beispiel: `regex: "power-state.+"`
item	Dieses Schlüsselwort ermöglicht die Suche innerhalb der von `regex` gefundenen Zeilen. Wenn kein regulärer Ausdruck bereitgestellt wurde, werden alle Zeilen getestet. Beispiel: `regex: "power"`
context	Dieses Schlüsselwort ermöglicht die Suche nach bestimmtem Inhalt. Ein `context` wird durch eine linksbündige Zeile definiert, der Zeilen mit vorangestellten Leerzeichen folgen.

	Beispiel:
	`context: "line .*"`
	
	Folgendes Beispiel ist ein Beispiel eines Konfigurationselements, das durch Nutzung von `context` geprüft werden könnte:
	
	`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>`
	
	Der oben stehende Test gewährleistet, dass „`ip address dhcp-bootp`“ auf `context "vlan 1"` gesetzt wird.

`min_occurrences`	Mit diesem Schlüsselwort können Sie eine Mindestanzahl von Testfällen festlegen. Beispiel: `min_occurrences: 3`
`max_occurrences`	Wie **`min_occurrences`**, aber mit Höchst- statt Mindestwert.
`required`	Mit diesem Schlüsselwort können Sie angeben, ob eine Übereinstimmung beim Test erforderlich ist. Der Wert des erforderlichen Felds kann YES („Ja“), NO („Nein“), ENABLED („Aktiviert“) oder DISABLED („Deaktiviert“) sein. Beispiel: `required: YES`

Referenz zu Compliancedateien für FireEye-Audits

Das FireEye-Audit basiert auf der Produktdokumentation von FireEye und Richtlinien zu gängigen Kriterien. Das Audit umfasst Tests für Überprüfung, Identifizierung und Authentifizierung, Anwendungsverwaltung, IPMI (Intelligent Platform Management Interface), aktivierten Diensten, Verschlüsselung und Konfiguration von Malwareerkennungssystemen. Gültige SSH-Anmeldedaten für einen „root“-Benutzer oder Administrator mit Vollzugriff werden benötigt.



Testtypen

FireEye-Compliancetests verwenden einen von drei Testtypen. Standardmäßig wird die folgende Testsyntax verwendet:

```
<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>
```

Schlüsselwörter

Die folgende Tabelle gibt an, wie die einzelnen Schlüsselwörter in FireEye-Compliancetests verwendet werden können:

Schlüsselwort	Einsatzbeispiel und unterstützte Einstellungen
type	CONFIG_CHECK CONFIG_CHECK_NOT RANDOMNESS_CHECK
description	Dieses Schlüsselwort bietet die Möglichkeit, dem ausgeführten Test eine kurze Beschreibung hinzuzufügen. Es wird empfohlen, dem Feld einen eindeutigen Wert zuzuweisen und sicherzustellen, dass andere Tests keinen identischen Eintrag in diesem Feld aufweisen. Dies ist erforderlich, weil SecurityCenter auf der Basis des Feldes `description` automatisch eine Plugin-ID generiert.

	Beispiel: `description: " Verify login authentication"`
info	Das Schlüsselwort ermöglicht es Benutzern, dem ausgeführten Test eine ausführlichere Beschreibung hinzuzufügen. Es sind mehrere `info`-Felder ohne vordefinierte Einschränkung zulässig. Der Inhalt eines `info`-Feldes sollte in doppelte Anführungszeichen gesetzt werden. Beispiel: `info: "Verifies login authentication configuration."`
see_also	Dieses Schlüsselwort ermöglicht es Benutzern, Links aufzunehmen, die ggf. hilfreiche Angaben zu einem Test enthalten. Beispiel: `see_also: "http://www.fireeye.com/support/"`
reference	Dieses Schlüsselwort ermöglicht den Zusatz von Querverweisen für Audittests. Beispiel: `reference: "PCI|2.2.3,SANS-CSC|1"`
solution	Das Schlüsselwort stellt Text zur Aufnahme von Textlösungen zur Compliance-Fehlerbehebung bereit. Beispiel: `solution: "Modify the configuration to add missing line"`
severity	Mit diesem Schlüsselwort können Benutzer die Testintensität festlegen. Die Intensität kann auf HIGH („Hoch“), MEDIUM („Moderat“) oder LOW („Niedrig“) festgelegt werden. Beispiel: `severity: MEDIUM`
regex	Mit diesem Schlüsselwort aktivieren Sie die Aufzählung von Elementen, die einem bestimmten regulären Ausdruck entsprechen. Wenn das Schlüsselwort „**regex**“ für einen Test festgelegt, aber keines der Tags „**expect**“ oder „**not_expect**“ angegeben wurde, meldet der Test alle diesem regulären Ausdruck entsprechenden Elemente. Beispiel: `regex: "power-state.+"`
expect	Dieses Schlüsselwort ermöglicht die Suche innerhalb der von `regex` gefundenen Zeilen. Alle von `regex` gefundenen Zeilen müssen mit der Einstellung `expect` übereinstimmen, damit der Test bestanden ist. Wenn kein regulärer Ausdruck angegeben wurde, werden alle Zeilen getestet. Es braucht jedoch nur eine Zeile gefunden zu werden. Beispiel: `regex: "power"`
not_expect	Vergleichbar mit **expect**, nur ist der Test bei Übereinstimmung nicht bestanden. Werden sowohl **expect** als auch **not_expect** weggelassen, werden alle zutreffenden Zeilen als Infomeldung angezeigt.
min_occurrences	Mit diesem Schlüsselwort können Sie eine Mindestanzahl von Testfällen festsetzen.

	Beispiel: `min_occurrences: 3`
max_occurrences	Wie **min_occurrences**, aber mit Höchst- statt Mindestwert.
required	Mit diesem Schlüsselwort können Sie angeben, ob eine Übereinstimmung beim Test erforderlich ist. Der Wert des erforderlichen Felds kann YES, NO, ENABLED oder DISABLED sein. Beispiel: `required: YES`

cmd	Hiermit können Benutzer einen „show“-Befehl ausführen.
	
	Beispiel:
	`cmd: "show version"`
	
	Es sind nur „**show**“-Befehle zulässig.
	
	`<item>`
	` type: CONFIG_CHECK`
	` cmd: "show version"`
	` description: "Show Product version"`
	` regex: "Proaduct model:"`
	` expect: "1234"`
	`</item>`

Referenz zu Compliancedateien für Datenbankkonfigurationsaudits

In diesem Abschnitt werden Format und Funktionen von Datenbankcompliancetests und das Prinzip hinter den einzelnen Einstellungen erläutert.

Verwendung von Anführungszeichen:

Um Auditfelder gesetzte einzelne bzw. doppelte Anführungszeichen werden austauschbar verwendet. Ausgenommen sind lediglich die folgenden Fälle:

1. Bei Windows-Compliancetests, in denen Sonderfelder (z. B. CRLF) literal interpretiert werden müssen, müssen einzelne Anführungszeichen gesetzt werden. Eingebettete Felder, die als Strings zu interpretieren sind, müssen mit Escapezeichen versehen werden.

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

2. Doppelte Anführungszeichen sind erforderlich, wenn „include_paths“ und „exclude_paths“ für WindowsFiles verwendet werden.

Wenn Strings in einem beliebigen Feldtyp (`description`, `value_data`, `regex` usw.) verwendet werden, die doppelte Anführungszeichen aufweisen, kann auf zweierlei Weise verfahren werden:

a. Verwenden Sie den jeweils anderen Anführungszeichentyp als äußere umschließende Anführungszeichen.

Beispiel:

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

b. Stellen Sie eingebetteten doppelten Anführungszeichen den umgekehrten Schrägstrich („\“) als Escapezeichen voran.

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

Testtyp

Alle Compliancetests für Datenbanken müssen in die Kapselung **check_type** gesetzt werden, die mit der Typangabe „Database“ zu versehen ist. Dies ist erforderlich, um **.audit**-Dateien für Datenbanken von anderen Compliancetests unterscheiden zu können. Das Feld **check_type** erfordert zwei zusätzliche Parameter:

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

Die folgenden Datenbanktypen können mit Audits überprüft werden:

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

Der Wert **version** ist gegenwärtig fest auf „1“ eingestellt.

Beispiel:

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

Schlüsselwörter

Die folgende Tabelle gibt an, wie die einzelnen Schlüsselwörter in Datenbankcompliancetests verwendet werden können:

Schlüsselwort	Einsatzbeispiel und unterstützte Einstellungen
type	`SQL_POLICY`
description	Dieses Schlüsselwort bietet die Möglichkeit, dem ausgeführten Test eine kurze Beschreibung hinzuzufügen. Es wird dringend empfohlen, dem Feld einen eindeutigen Wert zuzuweisen und sicherzustellen, dass andere Tests keinen identischen Eintrag im Feld aufweisen. Auf der Basis des Feldes generiert Tenable SecurityCenter automatisch eine eindeutige Plugin-ID. Beispiel: `description: "DBMS Password Complexity"`
info	Dieses Schlüsselwort wird verwendet, um dem ausgeführten Test eine ausführlichere Beschreibung hinzuzufügen. Dies könnte eine Rechtsvorschrift, eine URL, eine Unternehmensrichtlinie oder eine Angabe dafür sein, warum diese Einstellung erforderlich ist. Mehrere **info**-Felder lassen sich in separaten Zeilen hinzufügen, um den Text als Abschnitt zu formatieren. Die Anzahl der verwendbaren **info**-Felder ist nicht begrenzt.

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

sql_request	Mit diesem Schlüsselwort wird die eigentliche **SQL**-Anforderung ermittelt, die an die Datenbank übermittelt werden soll. Durch Verwendung kommagetrennter Anforderungs- und Rückgabewerte können auch Datenarrays aus einer **SQL**-Anforderung angefordert und zurückgegeben werden.
	
	Beispiel:
	`sql_request: "select name from sys.sql_logins where type = 'S' and is_policy_checked <> '1'"`
	
	Beispiel:
	`sql_request: "select name, value_in_use from sys.configurations where name = 'clr enabled'"`

sql_types	Für dieses Schlüsselwort gibt es zwei Optionen: **POLICY_VARCHAR** und **POLICY_INTEGER**. Verwenden Sie **POLICY_INTEGER** für numerische Werte zwischen 0 und 2147483647 sowie **POLICY_VARCHAR** für jeden anderen Rückgabewerttyp.
	
	Beispiel:
	`sql_types: POLICY_VARCHAR`
	
	Beispiel:
	`sql_types: POLICY_VARCHAR,POLICY_INTEGER`
	
	Konfigurieren Sie bei mehreren Rückgabeelementen **sql_types** in einer kommagetrennten Liste, die die Datentypen aller **SQL**-Rückgabeergebnisse aufnimmt. Das obige Beispiel zeigt, dass der erste Rückgabewert der **SQL**-Abfrage ein **varchar** und der zweite Rückgabewert eine ganze Zahl ist.

sql_expect	Mit diesem Schlüsselwort wird der erwartete Rückgabewert der **SQL**-Anforderung bestimmt. Hier kann ein exakter Wert einschließlich **NULL** oder „**0**“ erforderlich sein. Außerdem können für **POLICY_VARCHAR sql_types** auch reguläre Ausdrücke erforderlich sein.
	
	Beispiel:
	`sql_expect: regex:"^.+Failure" || regex:"^.+ALL"`
	
	Beispiel:
	`sql_expect: NULL`
	
	Beispiel:
	`sql_expect: 0 || "0"`
	
	Bei ganzzahligen Rückgabewerten sind doppelte Anführungszeichen optional.
	
	Beispiel:
	`sql_expect: "clr enabled",0`
	
	Ein Datenarray kann für eine **SQL**-Anfrage zurückgegeben und in einem kommagetrennten Format in das Feld **sql_expect** eingesetzt werden.

Beispiele für die Befehlszeile

In diesem Abschnitt sind mehrere Beispiele gängiger Audits enthalten, die in Datenbankcompliancetests verwendet werden. Das Befehlszeilenbinärprogramm **nasl** wird als Möglichkeit verwendet, Tests zwischendurch schnell und unkompliziert durchzuführen. Sie können alle nachfolgend gezeigten **.audit**-Dateien problemlos in Ihre Nessus 4 oder SecurityCenter-Scanrichtlinien integrieren. Allerdings sind für schnelle Audits eines einzelnen Systems Befehlszeilentests effizienter. Der Befehl wird jedes Mal wie folgt aus dem Verzeichnis **/opt/nessus/bin** heraus ausgeführt:

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

<IP> ist die IP-Adresse des geprüften Systems.

Abhängig vom Typ der zu prüfenden Datenbank werden Sie zur Eingabe weiterer Parameter zusätzlich zur verwendeten Auditdatei aufgefordert. Bei Oracle-Audits beispielsweise müssen Sie die Datenbank-SID und den Oracle-Anmeldetyp angeben:

```
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
```

Wenden Sie sich an Ihren Datenbankadministrator, um die richtigen Anmeldeparameter zu erhalten.

Beispiel 1: Nach Anmeldenamen ohne Ablaufdatum suchen

Die nachfolgend gezeigte einfache **.audit**-Datei sucht nach beliebigen SQL Server-Anmeldenamen ohne Ablaufdatum. Werden solche Anmeldenamen gefunden, dann wird das Audit als fehlgeschlagen gewertet und eine Liste der nicht konformen Anmeldenamen angezeigt.

```
<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>
```

Wenn Sie diesen Befehl ausführen, müsste auf einem System, das die Compliance-Anforderungen erfüllt, folgende Ausgabe erscheinen:

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

Compliance-Anforderungen sehen in der Regel vor, dass es für Datenbankanmeldenamen ein Ablaufdatum gibt.

Bei einem fehlgeschlagenen Audit würde die folgende Ausgabe zurückgegeben:

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

Database logins with no expiration date pose a security threat.

Remote value:

"distributor_admin"

Policy value:

NULL
```

Diese Ausgabe zeigt, dass für das Konto „distributor_admin“ kein Ablaufdatum konfiguriert wurde und dass dies in der Systemsicherheitsrichtlinie zu prüfen ist.

Beispiel 2: Aktivierungsstatus einer unautorisierten gespeicherten Prozedur testen

Bei diesem Audit wird geprüft, ob die gespeicherte Prozedur „SQL Mail XPs“ aktiviert ist. Externe gespeicherte Prozeduren können auf bestimmten Systemen ein Sicherheitsrisiko darstellen, weswegen häufig ihre Deaktivierung verlangt wird.

```
<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>
```

Beim obigen Test wird das Ergebnis „Passed“ zurückgegeben, wenn die gespeicherte Prozedur „SQL Mail XPs“ deaktiviert ist („value_in_use = 0“). Andernfalls wird das Ergebnis „Failed“ zurückgegeben.

Beispiel 3: Datenbankstatus mit gemischten Ergebnissen für sql_types testen

In einigen Fällen erfordern Datenbankabfragen für Compliancetests mehrere Datenanforderungen, deren Ergebnisse auch mehrere Datentypen aufweisen können. Das nachfolgende Beispiel ist ein Audit, in dem Datentypen gemischt werden. Es veranschaulicht, wie die Ausgabe geparst werden kann.

```
<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>
```

Beachten Sie, dass **sql_request**, **sql_types** und **sql_expect** kommagetrennte Werte enthalten.

Bedingungen

Sie können eine **if/then/else**-Logik in der Datenbankrichtlinie definieren. Dies ermöglicht es, dem Endbenutzer statt des Ergebnisses („Passed“/„Failed“) eine Warnung anzuzeigen, falls das Audit erfolgreich ist.

Die Syntax zur Ausführung von Bedingungen lautet wie folgt:

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

Beispiel:

```
<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>
```

Ob die Bedingung erfüllt wird oder nicht, wird im Bericht nicht angezeigt, da es sich hier um einen im Hintergrund stattfindenden Test handelt.

Bei den Bedingungen wird zwischen den Typen „**and**“ und „**or**“ unterschieden.

Referenz zu Compliancedateien für UNIX-Konfigurationsaudits

In diesem Abschnitt werden die integrierten Funktionen von UNIX-Compliancetests und das Prinzip hinter den einzelnen Einstellungen erläutert.

Verwendung von Anführungszeichen:

Um Auditfelder gesetzte einzelne bzw. doppelte Anführungszeichen werden austauschbar verwendet. Ausgenommen sind lediglich die folgenden Fälle:

1. Bei Windows-Compliancetests, in denen Sonderfelder (z. B. CRLF) literal interpretiert werden müssen, müssen einzelne Anführungszeichen gesetzt werden. Eingebettete Felder, die als Strings zu interpretieren sind, müssen mit Escapezeichen versehen werden.

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

2. Doppelte Anführungszeichen sind erforderlich, wenn „include_paths“ und „exclude_paths“ für „WindowsFiles“ verwendet werden.

Wenn Strings in einem beliebigen Feldtyp (`description`, `value_data`, `regex` usw.) verwendet werden, die doppelte Anführungszeichen aufweisen, kann auf zweierlei Weise verfahren werden:

a. Verwenden Sie den jeweils anderen Anführungszeichentyp als äußere umschließende Anführungszeichen.

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

b. Stellen Sie eingebetteten doppelten Anführungszeichen den umgekehrten Schrägstrich („\“) als Escapezeichen voran.

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

Testtyp

Alle UNIX-Compliancetests für Datenbanken müssen in die „**check_type**“-Kapselung gesetzt werden, die mit der Typangabe „Unix“ zu versehen ist. Anhang A enthält ein Beispiel für einen UNIX-Compliancetest, der mit der **check_type**-Einstellung „Unix“ beginnt und mit dem Tag „**</check_type>**“ abgeschlossen wird.

Dies ist erforderlich, um **.audit**-Dateien für UNIX von Compliancetests für Windows oder andere Plattformen unterscheiden zu können.

Die Datei wird über SSH in einen Pufferspeicher auf dem Nessus-Server eingelesen. Dieser Puffer wird dann auf Compliance geprüft.

Schlüsselwörter

Die folgende Tabelle gibt an, wie die einzelnen Schlüsselwörter in UNIX-Compliancetests verwendet werden können:

Schlüsselwort	Einsatzbeispiel und unterstützte Einstellungen
attr	Dieses Schlüsselwort wird in Verbindung mit FILE_CHECK und FILE_CHECK_NOT verwendet, um die Dateiattribute einer Datei zu überprüfen. Details zur Konfiguration der Dateiattribute einer Datei entnehmen Sie der Manpage zu „chattr(1)“.
comment	In dieses Feld können Sie zusätzliche Informationen eingeben, die nicht in das Feld `description` passen. Beispiel: `comment: (CWD - Current working directory)`
description	Dieses Schlüsselwort bietet die Möglichkeit, dem ausgeführten Test eine kurze Beschreibung hinzuzufügen. Es wird empfohlen, dem Feld einen eindeutigen Wert zuzuweisen und sicherzustellen, dass andere Tests keinen identischen Eintrag in diesem Feld aufweisen. Auf der Basis des Feldes **description** generiert Tenable SecurityCenter automatisch eine eindeutige Plugin-ID. Beispiel: `description: "Permission and ownership check for /etc/at.allow"`
dont_echo_cmd	Dieses Schlüsselwort, das bei UNIX-Compliancetests des Typs „CMD_EXEC“ verwendet wird, weist das Audit an, den beim Test ausgeführten Befehl in der Ausgabe wegzulassen. Es werden nur die Resultate des Befehls angezeigt. Beispiel: `dont_echo_cmd: YES`

except	Mit diesem Schlüsselwort können Sie bestimmte Benutzer, Dienste und Dateien aus dem Test ausschließen.
	
	Beispiel:
	`except: "guest"`
	
	Es können auch mehrere Benutzerkonten durch Pipe-Zeichen getrennt aufgelistet werden.
	
	Beispiel:
	`except: "guest" | "guest1" | "guest2"`

expect	Dieses Schlüsselwort wird in Verbindung mit **regex** verwendet. Es ermöglicht das Suchen nach bestimmten Werten innerhalb von Dateien.
	
	Beispiel:
	`<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	Mit diesem Schlüsselwort geben Sie den absoluten oder relativen Pfad einer Datei an, deren Berechtigungen und Besitzeinstellungen getestet werden sollen. Beispiele: file: "/etc/inet/inetd.conf" file: "~/inetd.conf" Der Wert von **file** kann auch ein Glob sein. Beispiel: file: "/var/log/*" Diese Funktion ist besonders nützlich, wenn alle Dateien im jeweiligen Verzeichnis mit „FILE_CHECK“, „FILE_CONTENT_CHECK“, „FILE_CHECK_NOT“ oder „FILE_CONTENT_CHECK_NOT“ auf Berechtigungen oder Inhalte geprüft werden sollen.
file_type	Mit diesem Schlüsselwort wird der Typ der gesuchten Datei beschrieben. Die folgende Liste enthält die unterstützten Dateitypen. • b: Block-Special-Dateien (gepuffert) • c: Character-Special-Dateien (ungepuffert) • d: Verzeichnis • p: Named Pipe (FIFO) • f: reguläre Datei Beispiel: file_type: "f" Ein oder mehrere Dateitypen können – durch Pipe-Zeichen getrennt – im selben String aufgeführt sein. Beispiel: file_type: "c|b"
group	Mit diesem Schlüsselwort wird die Gruppe einer Datei angegeben. Es wird immer in Verbindung mit dem Schlüsselwort **file** verwendet. Das Schlüsselwort **group** kann auch den Wert „none“ haben und ermöglicht in einem solchen Fall die Suche nach Dateien ohne Besitzer. Beispiel: group: "root" Die Gruppe kann auch mit einer logischen ODER-Bedingung angegeben werden. Hierzu wird die folgende Syntax verwendet: group: "root" || "bin" || "sys"
ignore	Mit diesem Schlüsselwort wird festgelegt, dass die angegebenen Dateien bei der Suche ignoriert werden. Das Schlüsselwort ist für die Testtypen „FILE_CHECK“, „FILE_CONTENT_CHECK“, „FILE_CHECK_NOT“ und

	„FILE_CONTENT_CHECK_NOT“ verfügbar.
	
	Beispiele:
	`# 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	Dieses Schlüsselwort wird verwendet, um dem ausgeführten Test eine ausführlichere Beschreibung hinzuzufügen. Dies könnte eine Rechtsvorschrift, eine URL, eine Unternehmensrichtlinie oder eine Angabe dafür sein, warum diese Einstellung erforderlich ist. Mehrere **info**-Felder lassen sich in separaten Zeilen hinzufügen, um den Text als Abschnitt zu formatieren. Die Anzahl der verwendbaren **info**-Felder ist nicht begrenzt. Beispiel: `info: "ref. CIS_AIX_Benchmark_v1.0.1.pdf ch 1, pg 28-29."`
levels	Dieses Schlüsselwort wird in Verbindung mit CHKCONFIG verwendet, um die Runlevels anzugeben, für die ein Dienst ausgeführt werden muss. Alle Runlevels müssen in einem einzigen String beschrieben werden. Wenn beispielsweise der Dienst „sendmail“ auf den Runlevels 1, 2 und 3 ausgeführt werden muss, lautet der entsprechende **levels**-Wert im CHKCONFIG-Test wie folgt: `levels: "123"`
mask	Dieses Schlüsselwort ist das Gegenteil von **mode**. Hiermit können Berechtigungen angegeben werden, die für einen bestimmten Benutzer, eine Gruppe oder ein anderes Mitglied **nicht** verfügbar sein sollen. Anders als **mode**, mit dem auf *exaktes* Vorhandensein eines Berechtigungswerts getestet wird, sind **mask**-Audits weiter gefasst und überprüfen, ob sich eine Datei oder ein Verzeichnis auf einer Ebene befindet, die mindestens so sicher ist wie die durch **mask** angegebene. (Im Gegensatz dazu würde bei **mode** eine Datei mit dem Berechtigungswert 640 zu einem fehlgeschlagenen Auditergebnis führen, da dieser Wert nicht exakt 644 entspricht; **mask** hingegen würde 640 als „sicherer“ als 644 erkennen und das Audit als erfolgreich werten.) Beispiel: `mask: 022` Hiermit würde festgelegt, dass jede Berechtigung für den Besitzer akzeptabel ist, während Schreibberechtigungen für Gruppen und andere Mitglieder nicht vorhanden sein dürfen. Der Wert „7“ für **mask** würde vorsehen, dass es keine Berechtigungen für den betreffenden Besitzer, die Gruppe oder andere Mitglieder gibt.

md5	Dieses Schlüsselwort wird in „FILE_CHECK“- und „FILE_CHECK_NOT“-Tests verwendet, um sicherzustellen, dass der MD5-Wert einer Datei wie von den Richtlinien vorgesehen festgelegt ist.
	
	Beispiel:
	`<custom_item>`
	`type: FILE_CHECK`

	description: "/etc/passwd has the proper md5 set" required: YES file: "/etc/passwd" md5: "ce35dc081fd848763cab2cfd442f8c22" </custom_item>
mode	Mit diesem Schlüsselwort wird der Berechtigungssatz für eine Datei oder einen Ordner beschrieben. Der Wert des Schlüsselworts **mode** kann als String oder im Oktalformat angegeben werden. Beispiele: mode: "-rw-r--r--" mode: "644" mode: "7644"
name	Dieses Schlüsselwort wird zur Angabe von Prozessnamen bei PROCESS_CHECK-Tests verwendet. Beispiel: name: "syslogd"
operator	Dieses Schlüsselwort wird in Verbindung mit „RPM_CHECK“ und „PKG_CHECK“ verwendet, um die Bedingung für einen erfolgreichen oder fehlgeschlagenen Test basierend auf der Version des installierten RPM-Pakets anzugeben. Die folgenden Werte lassen sich festlegen: • **lt** (kleiner als) • **lte** (kleiner oder gleich) • **gte** (größer oder gleich) • **gt** (größer als) • **eq** (gleich) Beispiel: operator: "lt"
owner	Mit diesem Schlüsselwort wird der Besitzer einer Datei angegeben. Es wird immer in Verbindung mit dem Schlüsselwort **file** verwendet. Das Schlüsselwort **owner** kann auch den Wert „none“ („kein“) haben und ermöglicht in einem solchen Fall die Suche nach Dateien ohne Besitzer. Beispiel: owner: "root" Der Besitz kann auch mit einer logischen ODER-Bedingung angegeben werden. Hierzu wird die folgende Syntax verwendet: owner: "root" || "bin" || "adm"
reference	Mit diesem Schlüsselwort können Sie Querverweise in die **.audit**-Datei setzen. Das Format ist „ref|ref-id1,ref|ref-id2“. Beispiel: 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	Mit diesem Schlüsselwort aktivieren Sie die Suche nach einer Datei, die einem bestimmten regulären Ausdruck entspricht.

	Beispiel: `regex: ".*LogLevel=9$"` Die folgenden Metazeichen erfordern einen gesonderten Umgang: + \ * ( ) ^ Wenn diese Zeichen literal ausgewertet werden sollen, müssen Sie sie entweder mit zwei umgekehrten Schrägstrichen („\\“) als Escapezeichen auszeichnen oder sie in eckige Klammern („[]“) setzen. Die folgenden Zeichen hingegen müssen für eine literale Auswertung mit nur einem umgekehrten Schrägstrich ausgezeichnet werden: . ? " ' . Dies hat mit der Art und Weise zu tun, wie der Compiler diese Zeichen auswertet.
required	Mit diesem Schlüsselwort wird angegeben, ob das geprüfte Element auf dem Remotesystem vorhanden sein muss oder nicht. Wenn **required** beispielsweise den Wert „NO“ hat und für **type** „FILE_CHECK“ festgelegt wurde, ist der Test erfolgreich, wenn die Datei vorhanden ist und die Berechtigungen den in der **.audit**-Datei angegebenen entsprechen, oder wenn die Datei nicht vorhanden ist. Hätte **required** hingegen den Wert „YES“, dann würde der obige Test fehlschlagen.

rpm	Mit diesem Schlüsselwort wird bei Verwendung von RPM_CHECK der zu suchende RPM angegeben.
	
	Beispiel:
	`<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	Mit diesem Schlüsselwort können Sie durchsuchbare Speicherorte auf einem Dateisystem angeben.
	
	Beispiel:
	`search_locations: "/bin"`
	
	Es können auch mehrere Suchpositionen durch Pipe-Zeichen getrennt aufgelistet werden.
	
	Beispiel:
	`search_locations: "/bin" | "/etc/init.d" | "/etc/rc0.d"`

see_also	Dieses Schlüsselwort lässt die Angabe von Links zu Referenzmaterial zu. Beispiel: `see_also: "https://benchmarks.cisecurity.org/tools2/linux/CIS_Redhat_Linux_5_Benchmark_v2.0.0.pdf"`

service	Dieses Schlüsselwort wird in Verbindung mit CHKCONFIG, XINETD_SVC und SVC_PROP verwendet, um den Dienst anzugeben, für den das Audit ausgeführt wird.
	
	Beispiel:
	`<custom_item>`
	`type: CHKCONFIG`

	description: "2.1 Disable Standard Services – Check if cups is disabled" service: "cups" levels: "123456" status: OFF </custom_item>
severity	Jedem **<item>** oder **<custom_item>** Test kann ein „**severity**“-Flag hinzugefügt werden. Dieses kann die Werte „LOW“ (niedrig), „MEDIUM“ (moderat) oder „HIGH“ (hoch) annehmen. Standardmäßig werden Ergebnisse, die die Compliance-Anforderungen nicht erfüllen, mit dem Schweregrad „high“ angegeben. Beispiel: severity: MEDIUM
solution	Dieses Schlüsselwort ermöglich den Zusatz von „Solution“ (Lösungstext) sofern verfügbar. Beispiel: solution: "Remove this file, if its not required"
status	Dieses Schlüsselwort wird in Verbindung mit PROCESS_CHECK, CHKCONFIG und XINETD_SVC verwendet, um zu bestimmen, ob ein Dienst, der auf einem bestimmten Host ausgeführt wird, ausgeführt werden darf oder deaktiviert werden soll. Das Schlüsselwort **status** kann zwei Werte annehmen: „ON“ (der Dienst darf ausgeführt werden) und „OFF“ (der Dienst wird deaktiviert). Beispiel: status: ON status: OFF
system	Mit diesem Schlüsselwort wird der Typ des Systems angegeben, auf dem der Test ausgeführt wird.  Das Schlüsselwort „**system**“ ist nur in Verbindung mit „**custom_item**“-Tests einsetzbar, nicht jedoch mit integrierten „**item**“-Tests. Die möglichen Werte sind diejenigen, die vom Befehl „**uname**“ unter dem Zielbetriebssystem zurückgegeben werden. Bei Solaris etwa heißt der Wert „SunOS“, bei Mac OS X „Darwin“, bei FreeBSD „FreeBSD“ usw. Beispiel: system: "SunOS"
timeout	Dieses Schlüsselwort wird in Verbindung mit CMD_EXEC verwendet. Es gibt an, wie lange (in Sekunden) der angegebene Befehl ausgeführt werden darf, bevor ein Timeout auftritt. Das Schlüsselwort ist nützlich in Fällen, in denen für die Ausführung eines bestimmten Befehls – z. B. des UNIX-Befehls „**find**“ – eine gewisse Zeit benötigt wird. Ist das Schlüsselwort nicht angegeben, dann tritt bei CMD_EXEC-Audits nach fünf Minuten ein Timeout auf. Beispiel: 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	Das Schlüsselwort **value** ist praktisch, um zu überprüfen, ob eine Einstellung auf dem System mit dem in der Richtlinie vorgegebenen Wert übereinstimmt. Beispiel: value: "90..max" Das Schlüsselwort **value** kann als Bereich angegeben werden ([zahl..max]). Wenn der Wert zwischen der angegebenen Zahl und „max“ liegt, ist der Test erfolgreich.

Benutzerdefinierte Elemente

Ein benutzerdefiniertes Element ist ein vollständiger Test, der auf der Basis der oben definierten Schlüsselwörter definiert wird. Die folgende Liste enthält die benutzerdefinierten Elemente. Jeder Test beginnt mit dem Tag „**<custom_item>**“ und endet mit „**</custom_item>**“. In die Tags eingeschlossen ist eine Liste mit einem oder mehreren Schlüsselwörtern, die vom Compliancetest-Parser zur Durchführung des Tests interpretiert werden.

Bei den benutzerdefinierten Audittests können „**</custom_item>**“ und „**</item>**“ als schließende Tags beliebig gegeneinander ausgetauscht werden.

AUDIT_XML

Der Audittest „AUDIT_XML“ ermöglicht Untersuchung und Prüfung von Inhalten einer XML-Datei. Hierzu werden zunächst XSL-Transformationen angewendet und die relevanten Daten extrahiert; danach erfolgt die Bestimmung der Compliance beruhend auf den Schlüsselwörtern **regex**, **expect** und **not_expect** (weitere Informationen finden Sie in Anhang C). Der Test umfasst mindestens vier Schlüsselwörter, den Schlüsselworttyp, eine Beschreibungsdatei und xsl_stmt-Anweisungen (obligatorisch), auf die die **regex**, **expect** oder **not_expect** Schlüsselwörter zur Prüfung des Inhalts folgen.

Beispiel:

```
<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>
```

Beachten Sie, dass beim Dateischlüsselwort Platzhalter zulässig sind. Beispiel:

```
<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

Der Audittest „CHKCONFIG“ ermöglicht eine Interaktion mit dem Utility „**chkconfig**“ auf dem Red Hat-Remotesystem, auf dem das Audit ausgeführt wird. Dieser Test umfasst die fünf obligatorischen Schlüsselwörter **type**, **description**, **service**, **levels** und **status**.

Ein CHKCONFIG-Audit funktioniert nur auf Red Hat-Systemen oder davon abgeleiteten Systemen wie Fedora.

Beispiel:

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

CMD_EXEC

Es ist möglich, auf dem Remotehost Befehle auszuführen und dann zu überprüfen, ob die Ausgabe den Erwartungen entspricht. Tests dieses Typs sollten mit extremer Vorsicht ausgeführt werden, da sie zwischen verschiedenen UNIX-Derivaten nicht immer portierbar sind.

Das Schlüsselwort **quiet** weist Nessus an, die Ausgabe des fehlgeschlagenen Befehls **nicht** anzuzeigen. Es kann auf „YES“ oder „NO“ festgelegt werden. Standardmäßig ist es auf „NO“ festgelegt, d. h., das Befehlsergebnis wird angezeigt. Ähnlich schränkt das Schlüsselwort „**dont_echo_cmd**“ die Ausgabe der Befehlsergebnisse (nicht jedoch den Befehl selbst) ein.

Mit dem Schlüsselwort **nosudo** kann Nessus angewiesen werden, „sudo“ **nicht** zur Ausführung von Befehlen zu verwenden. Hierzu wird der Wert „YES“ festgelegt. Standardmäßig ist das Schlüsselwort auf „NO“ festgelegt, d. h., sofern „sudo“ hierfür konfiguriert ist, wird es auch stets verwendet.

Beispiel:

```
<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-Compliance-Audits prüfen in der Regel auf das Vorhandensein einer Datei und die darin enthaltenen Einstellungen. Das Audit „FILE_CHECK“ verwendet mindestens vier Schlüsselwörter, um die Spezifikation dieser Tests zu ermöglichen. Die Schlüsselwörter **`type`**, **`description`** und **`file`** sind obligatorisch. Ihnen folgt mindestens ein Test. Die aktuelle Syntax unterstützt das Überprüfen der Berechtigungen von Besitzer, Gruppe und Datei.

Sie können in FILE_CHECK auch Globs verwenden (z. B. **`/var/log/*`***). Beachten Sie jedoch, dass Globs nur auf Dateien, nicht aber auf Verzeichnisse erweitert werden. Wenn ein Glob angegeben ist und mindestens eine Datei bei der Suche ignoriert werden soll, verwenden Sie das Schlüsselwort „**`ignore`**“, um die zu ignorierenden Dateien anzugeben.

Zulässige Schlüsselwörter sind:

```
uid: Numerische Benutzer-ID (z. B. 0)
gid: Numerische Gruppen-ID (z. B. 500)
check_uneveness: YES
system: Systemtyp (z. B. Linux)
description: Textbeschreibung des Dateitests
file: Vollständiger Pfad und Datei für den Test (z. B. /etc/sysconfig/sendmail)
owner: Eigentümer der Datei (z. B. root)
group: Gruppeneigentümer der Datei (z. B. bin)
mode: Berechtigungsmodus (z. B. 644)
mask: Dateimaske (z. B. 133)
md5: MD5-Hash einer Datei (z. B. 88d3dbe3760775a00b900a850b170fcd)
ignore: Datei, die ignoriert werden soll (z. B. /var/log/secure)
attr: Dateiattribut (z. B. ----i--------)
```

Dateiberechtigungen sind als unausgewogen zu betrachten, wenn „**`group`**“ oder „**`other`**“ weitergehende Berechtigungen haben als „**`owner`**“, oder wenn „**`other`**“ weitergehende Berechtigungen hat als „**`group`**“.

Hier folgen einige Beispiele:

```
<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

Das „FILE_CHECK_NOT“-Audit umfasst mindestens drei Schlüsselwörter. Die Schlüsselwörter **`type`**, **`description`** und **`file`** sind obligatorisch. Ihnen folgt mindestens ein Test. Die aktuelle Syntax unterstützt das Überprüfen der Berechtigungen von Besitzer, Gruppe und Datei. Ähnlich wie beim FILE_CHECK-Audit können mithilfe des Schlüsselworts „**`ignore`**“ Dateien angegeben werden, die ignoriert werden sollen, wenn ein Datei-Glob angegeben wurde.

Diese Funktion ist das Gegenteil von FILE_CHECK. Eine Richtlinie schlägt fehl, wenn eine Datei nicht vorhanden ist, oder wenn ihr Modus dem im Test selbst definierten Modus entspricht.

Sie können in FILE_CHECK_NOT auch Globs verwenden (z. B. **`/var/log/*`***). Beachten Sie jedoch, dass Globs nur auf Dateien, nicht aber auf Verzeichnisse erweitert werden.

Hier folgen einige Beispiele:

```
<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

Wie beim Testen auf Vorhandensein einer Datei und von Dateieinstellungen kann auch der Inhalt von Textdateien analysiert werden. Mithilfe regulärer Ausdrücke können Speicherorte auf vorhandene Inhalte überprüft werden. Mit dem Schlüsselwort „**ignore**“ können Sie Dateien an den angegebenen Speicherorten ignorieren.

Mithilfe des Feldes **string_required** kann angegeben werden, ob die im Rahmen des Audits gesuchte Zeichenfolge vorhanden sein muss oder nicht. Wenn diese Option nicht festgelegt ist, muss sie vorhanden sein. Mithilfe des Feldes **file_required** kann angegeben werden, ob die überwachte Datei vorhanden sein muss oder nicht. Wenn diese Option nicht festgelegt ist, muss sie vorhanden sein.

Hier folgen einige Beispiele:

```
<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>
```

Durch Hinzufügen einer Tilde („~“) zu einem Dateiparameter kann festgelegt werden, dass die Homeverzeichnisse des Benutzers von FILE_CONTENT_CHECK auf unzulässige Inhalte geprüft werden.

```
<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

Dieses Audit prüft die Inhalte einer Datei auf Übereinstimmung mit dem regulären Ausdruck im Feld **`regex`**. Die Funktion negiert FILE_CONTENT_CHECK, d. h., wenn der reguläre Ausdruck eine **Übereinstimmung** in der Datei hat, ist die Richtlinie nicht erfolgreich. Mit dem Schlüsselwort „**`ignore`**“ können Sie Dateien an den angegebenen Speicherorten ignorieren.

Mit diesem Richtlinienelement wird geprüft, ob die Datei den regulären Ausdruck „`regex`“ enthält und dieser Ausdruck nicht mit dem Feld „**`expect`**“ übereinstimmt.

Der zulässige Typ ist:

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

Sowohl **`regex`** als auch **`expect`** müssen für diesen Test angegeben werden.

Beispiel:

```
<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

Der Audittest „GRAMMAR_CHECK“ untersucht den Inhalt einer Datei und führt einen Vergleich mit einer weit gefassten „Grammatik“ durch (diese besteht aus mindestens einem regulären Ausdruck). Wenn **eine** Zeile in der Zieldatei keinem der regulären Ausdrücke entspricht, schlägt der Test fehl.

Beispiel:

```
<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

Der Audittest „MACOSX_DEFAULTS_READ“ prüft die Standardsystemwerte unter Mac OS X. Der Test verhält sich abhängig von bestimmten Eigenschaftseinstellungen unterschiedlich.

Wenn **plist_user** auf „all“ festgelegt ist, werden alle Benutzereinstellungen getestet, andernfalls wird nur die angegebene Benutzereinstellung getestet.

Wenn zusätzlich zur festgelegten Eigenschaft **plist_user** die Eigenschaft **byhost** auf YES festgelegt ist, wird auch folgende Abfrage ausgeführt:

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

Ist die Eigenschaft **plist_user**, nicht aber die Eigenschaft **byhost** festgelegt, dann wird folgende Abfrage ausgeführt:

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

Sind weder **byhost** noch **plist_user** festgelegt, dann wird folgende Abfrage ausgeführt:

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

Es werden die folgenden Eigenschaften unterstützt:

```
plist_name : Abzufragende plist. z. B. com.apple.digihub
plist_item : Zu testendes plist-Element, z. B. com.apple.digihub.blank.cd.appeared
plist_option: CANNOT_BE_NULL. Wenn diese Eigenschaft auf CANNOT_BE_NULL festgelegt ist,
wird der Test nicht bestanden, wenn die zu testende Einstellung nicht festgelegt wurde.
byhost: YES. Wird byhost auf YES festgelegt, dann erhalten Sie eine leicht abweichende
Abfrage.
```

Beispiele:

```
<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

Der Audittest „PKG_CHECK“ führt **`pkgchk`** für ein SunOS-System aus. Mithilfe des Schlüsselwortes **`pkg`** wird das zu suchende Paket angegeben, während das Schlüsselwort **`operator`** die Bedingung für einen erfolgreichen oder fehlgeschlagenen Test basierend auf der Version des installierten Pakets angibt.

Beispiele:

```
<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

Wie bei den Dateitests kann eine geprüfte UNIX-Plattform auch auf laufende Prozesse getestet werden. Die Implementierung führt den Befehl „**chkconfig -list**“ aus, um eine Liste laufender Prozesse abzurufen.

Beispiele:

```
<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

Der Audittest „RPM_CHECK“ dient der Überprüfung der Versionsnummern auf dem Remotesystem installierter RPM-Pakete. Dieser Test umfasst die vier obligatorischen Schlüsselwörter **type**, **description**, **rpm** und **operator** sowie das optionale Schlüsselwort **required**. Mithilfe des Schlüsselwortes **rpm** wird das zu suchende Paket angegeben, während das Schlüsselwort **operator** die Bedingung für einen erfolgreichen oder fehlgeschlagenen Test basierend auf der Version des installierten RPM-Pakets angibt.

Die Verwendung von RPM-Tests kann nicht auf Linux-Distributionen portiert werden. Aus diesem Grund ist RPM_CHECK insgesamt nicht portierbar.

Hier einige Beispiele (es wird dabei vorausgesetzt, dass **iproute-2.4.7-10** installiert ist):

```
<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

Der Audittest „SVC_PROP“ ermöglicht eine Interaktion mit dem Tool „**`svcprop -p`**“ auf einem Solaris 10-System. Hiermit können Sie Eigenschaften abfragen, die einem bestimmten Dienst zugeordnet sind. Mit dem Schlüsselwort **`service`** wird der zu prüfende Dienst angegeben. Das Schlüsselwort **`property`** gibt den Namen der Eigenschaft an, die abgefragt werden soll. Das Schlüsselwort **`value`** ist der erwartete Wert der Eigenschaft. Der erwartete Wert kann auch ein regulärer Ausdruck sein.

Mithilfe des Feldes **`svcprop_option`** kann angegeben werden, ob die im Rahmen des Audits gesuchte Zeichenfolge vorhanden sein muss oder nicht. Die Argumente des Feldes sind CAN_BE_NULL und CANNOT_BE_NULL.

Beispiele:

```
<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

Mit dem Audittest „XINETD_SVC“ lässt sich der Startstatus von xinetd-Diensten prüfen. Dieser Test umfasst die vier obligatorischen Schlüsselwörter **`type`**, **`description`**, **`service`** und **`status`**.

Ein solches Audit funktioniert nur auf Red Hat-Systemen oder davon abgeleiteten Systemen wie Fedora.

Beispiel:

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

Integrierte Tests

Anwendungsfälle, die durch die oben beschriebenen Tests nicht abgedeckt sind, müssen als benutzerdefinierte Namen in NASL geschrieben werden. Alle derartigen Tests fallen in die Kategorie „built-in“ (integriert). Jeder Test beginnt mit dem Tag „**`<item>`**“ und endet mit „**`</item>`**“. In die Tags eingeschlossen ist eine Liste mit einem oder mehreren Schlüsselwörtern, die vom Compliancetest-Parser zur Durchführung des Tests interpretiert werden. Die folgende Liste enthält die vorhandenen Tests.

Das Schlüsselwort „**`system`**“ ist für integrierte Tests nicht verfügbar; wird es verwendet, dann wird ein Syntaxfehler ausgelöst.

Kennwortverwaltung

In den folgenden Beispielen werden <min> und <max> zur Darstellung eines ganzzahligen Wertes (und nicht eines Strings) in den Auditwerten verwendet.

In Fällen, in denen die Minimal- oder Maximalwerte nicht bekannt sind, setzen Sie die Strings „Min“ bzw. „Max“ statt des ganzzahligen Werts.

min_password_length

Syntax

```
<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>
```

Dieser integrierte Test stellt sicher, dass die Mindestlänge des Kennworts auf dem Remotesystem im Bereich „**<min>..<max>**“ liegt. Die Festlegung einer Mindestlänge für das Kennwort zwingt Benutzer zur Definition komplexerer Kennwörter.

Betriebssystem	Implementierung
Linux	Die Mindestlänge für Kennwörter ist in **/etc/login.defs** als PASS_MIN_LEN definiert.
Solaris	Die Mindestlänge für Kennwörter ist in **/etc/default/passwd** als PASSLENGTH definiert. Beachten Sie, dass hiermit auch die Maximallänge für Kennwörter gesteuert wird.
HP-UX	Die Mindestlänge für Kennwörter ist in **/etc/default/security** als MIN_PASSWORD_LENGTH definiert.
Mac OS X	Die Mindestlänge für Kennwörter ist als „minChar“ in der lokalen Richtlinie definiert, die mit dem Befehl **pwpolicy** definiert wurde.

Beispiel:

```
 <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

Syntax

```
<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>
```

Diese integrierte Funktion gewährleistet, dass das maximale Kennwortalter (d. h. der Zeitraum, nach dessen Ablauf die Benutzer zum Ändern ihrer Kennwörter aufgefordert werden) innerhalb des definierten Bereichs liegt.

Die Definition eines maximalen Kennwortalters verhindert, dass Benutzer jahrelang dasselbe Kennwort einsetzen. Durch häufiges Wechseln von Kennwörtern wird so vermieden, dass ein Angreifer, der in den Besitz eines Kennworts gekommen ist, dieses unendlich lang verwenden kann.

Betriebssystem	Implementierung
Linux	Die Variable PASS_MAX_DAYS ist in **/etc/login.defs** definiert.
Solaris	Die Variable MAXWEEKS in **/etc/default/passwd** definiert die maximal zulässige Anzahl von Wochen, für die ein Kennwort verwendet werden darf.
HP-UX	Dieser Wert wird durch die Variable PASSWORD_MAXDAYS in **/etc/default/security** gesteuert.
Mac OS X	Die Option „maxMinutesUntilChangePassword“ der Kennwortrichtlinie (die mit dem Tool **pwpolicy** definiert wird) kann zur Festlegung dieses Wertes verwendet werden.

Beispiel:

```
 <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

Syntax

```
<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>
```

Diese integrierte Funktion gewährleistet, dass das minimale Kennwortalter (d. h. der Zeitraum, vor dessen Ablauf Benutzer ihre Kennwörter nicht ändern dürfen) innerhalb des definierten Bereichs liegt.

Die Festlegung eines Kennwortmindestalters verhindert, dass Benutzer ihre Kennwörter zu häufig ändern, weil sie die Kennwortchronik außer Kraft setzen möchten. Manche Benutzer tun dies, um ein bereits einmal verwendetes Kennwort erneut zu nutzen und so die Anforderungen an die Änderung von Kennwörtern zu umgehen.

Betriebssystem	Implementierung
Linux	Die Variable PASS_MIN_DAYS ist in **/etc/login.defs** definiert.
Solaris	Die Variable MINWEEKS in **/etc/default/passwd** definiert die minimal erforderliche Anzahl von Wochen, für die ein Kennwort verwendet werden darf.
HP-UX	Dieser Wert wird durch die Variable PASSWORD_MINDAYS in **/etc/default/security** gesteuert.
Mac OS X	Diese Option wird nicht unterstützt.

Beispiel:

```
<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>
```

Root-Zugriff
root_login_from_console

Syntax

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

Diese integrierte Funktion stellt sicher, dass der Benutzer „root“ sich nur direkt über die physische Konsole am Remotesystem anmelden kann.

Die Logik hinter diesem Test besteht darin, dass von der direkten Nutzung des Root-Kontos zu Administrationszwecken abgeraten wird, damit jeder Zugriff einer bestimmten Person zugeordnet werden kann. Verwenden Sie stattdessen ein generisches Benutzerkonto (das auf BSD-Systemen Mitglied der Serveradministrationsgruppe (wheel) sein muss) und nutzen dann „**`su`**“ (oder „**`sudo`**“), um die Berechtigungen für Administrationszwecke hochstufen zu können.

Betriebssystem	Implementierung
Linux und HP-UX	Vergewissern Sie sich, dass **`/etc/securetty`** vorhanden ist und nur „console“ enthält.
Solaris	Vergewissern Sie sich, dass **`/etc/default/login`** die Zeile „`CONSOLE=/dev/console`“ enthält.
Mac OS X	Diese Option wird nicht unterstützt.

Berechtigungsverwaltung
accounts_bad_home_permissions

Syntax

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

Diese integrierte Funktion stellt sicher, dass das Benutzerverzeichnis jedes nichtprivilegierten Benutzers dem Benutzer gehört, und dass Dritte, die zur selben Gruppe oder zur Gruppe „everyone“ gehören, keine Schreibrechte für dieses Verzeichnis haben. Es wird allgemein empfohlen, für Benutzerverzeichnisse den Modus 0755 oder einen noch strengeren Modus (z. B. 0700) festzulegen. Dieser Test ist erfolgreich, wenn alle Benutzerverzeichnisse korrekt konfiguriert sind; andernfalls schlägt er fehl. Mithilfe der Schlüsselwörter **`mode`** oder **`mask`** können hier die gewünschten Berechtigungsebenen für Benutzerverzeichnisse angegeben werden. Das Schlüsselwort **`mode`** akzeptiert Benutzerverzeichnisse, die exakt der angegebenen Ebene entsprechen, während das Schlüsselwort **`mask`** Benutzerverzeichnisse akzeptiert, die die angegebene oder eine noch sicherere Ebene aufweisen.

Wenn Dritte in ein Benutzerverzeichnis schreiben können, können sie die Ausführung beliebiger Befehle durch den Benutzer erzwingen, indem sie die Dateien **`~/.profile`**, **`~/.cshrc`**, **`~/.bashrc`** manipulieren.

Müssen Dateien von mehreren Benutzern derselben Gruppe bearbeitet werden können, dann wird in der Regel empfohlen, statt eines Benutzerverzeichnisses ein speziell angelegtes Verzeichnis zu verwenden, in das die Mitglieder der Gruppe schreiben können.

Führen Sie für fehlkonfigurierte Benutzerverzeichnisse „**`chmod 0755 <user directory>`**“ aus und ändern Sie den Besitzer entsprechend.

accounts_bad_home_group_permissions

Syntax

```
<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>
```

Diese integrierte Funktion ähnelt von der Funktionsweise her **`accounts_bad_home_permissions`**, doch wird hiermit sichergestellt, dass die primäre Gruppe des Benutzers auch Besitzer seiner Homeverzeichnisse als Gruppe ist.

accounts_without_home_dir

Syntax

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

Diese integrierte Funktion stellt sicher, dass alle Benutzer über ein Benutzerverzeichnis verfügen. Ist dies der Fall, dann ist der Test erfolgreich; andernfalls schlägt er fehl. Beachten Sie, dass Besitzer und Berechtigungen des Benutzerverzeichnisses bei diesem Test nicht geprüft werden.

Es wird grundsätzlich empfohlen, für jeden Benutzer auf einem System ein Benutzerverzeichnis zu definieren, weil es Tools gibt, die aus diesem Verzeichnis lesen oder in es schreiben müssen (beispielsweise prüft **`sendmail`** auf Vorhandensein der Datei **`~/.forward`**). Wenn ein Benutzer sich nicht anmelden muss, sollte stattdessen eine nicht vorhandene Shell (z. B. **`/bin/false`**) definiert werden. Auf vielen Systemen erhält ein Benutzer auch ohne Benutzerverzeichnis Anmelderechte, doch ist das Benutzerverzeichnis in diesem Fall „/“.

invalid_login_shells

Syntax

```
<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>
```

Diese integrierte Funktion stellt sicher, dass alle Benutzer über eine gültige Shell entsprechend der Definition in **/etc/shells** verfügen.

Die Datei **/etc/shells** wird von Anwendungen wie Sendmail und FTP-Servern verwendet, um festzustellen, ob eine Shell auf dem System gültig ist. Zwar wird diese Datei nicht vom Anmeldeprogramm verwendet, doch können Administratoren mit ihr definieren, welche Shells auf dem System gültig sind. Mithilfe des Tests **invalid_login_shells** kann überprüft werden, ob für alle Benutzer in der Datei **/etc/passwd** gültige Shells entsprechend der Definition in der Datei **/etc/shells** konfiguriert sind.

Hierdurch werden unzulässige Praktiken wie die Verwendung von **/sbin/passwd** als Shell verhindert, mit der Benutzer ihre Kennwörter ändern können. Wenn Sie die Anmeldung durch den Benutzer verhindern möchten, erstellen Sie eine ungültige Shell in **/etc/shells** (z. B. „**/nonexistent**“) und legen Sie diese für die gewünschten Benutzer fest.

Wenn Sie Benutzer ohne gültige Shell haben, definieren Sie eine gültige Shell für sie.

login_shells_with_suid

Syntax

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

Diese integrierte Funktion stellt sicher, dass keine Shell über „set-uid“-Fähigkeiten verfügt.

Eine „setuid“-Shell zeichnet sich dadurch aus, dass der Prozess beim Shell-Start seine Berechtigungen festsetzen kann (eine „setuid“-Shell für „root“ gewährt beispielsweise jedem Benutzer Superuser-Berechtigungen).

Das Vorhandensein einer „setuid“-Shell untergräbt den Zweck der Verwendung von UIDs und GIDs und macht die Zugriffssteuerung wesentlich komplexer.

Löschen Sie ggf. das gesetzte SUID-Bit aus allen Shells.

login_shells_writeable

Syntax

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

Diese integrierte Funktion stellt sicher, dass keine Shell Schreibrechte für alle (world-writeable) bzw. für Gruppen (group-writeable) aufweist.

Wenn eine Shell Schreibrechte für alle (oder für Gruppen) aufweist, können Benutzer ohne Berechtigungen sie durch beliebige Programme ersetzen. Auf diese Weise könnte ein Angreifer andere Benutzer dieser Shell dazu zwingen, nach dem Anmelden beliebige Befehle auszuführen.

Stellen Sie sicher, dass die Berechtigungen aller Shells wie erforderlich festgelegt sind.

login_shells_bad_owner

Syntax

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

Diese integrierte Funktion stellt sicher, dass jede Shell den Benutzern „root“ oder „bin“ zugeordnet ist.

Wie bei Shells mit ungültigen Berechtigungen kann ein Benutzer, der Besitzer einer von anderen Benutzern verwendeten Shell ist, diese modifizieren und so andere Benutzer dazu zwingen, beim Anmelden beliebige Befehle auszuführen.

Systemweit eingesetzte Binärdateien sollten nur von „root“ und/oder „bin“ geändert werden dürfen.

Kennwortdateiverwaltung

passwd_file_consistency

Syntax

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

Diese integrierte Funktion stellt sicher, dass jede Zeile in **`/etc/passwd`** ein gültiges Format aufweist (z. B. sieben Felder, die durch einen Doppelpunkt getrennt sind). Ist eine Zeile fehlerhaft formatiert, dann wird dies gemeldet, und der Test schlägt fehl.

Die Verwendung einer fehlformatierten Datei **`/etc/passwd`** kann dazu führen, dass verschiedene Tools zur Benutzerverwaltung nicht mehr funktionieren. Weiterhin kann dies ein Hinweis auf ein Eindringen oder einen Bug in einer selbstentwickelten Benutzerverwaltungsanwendung sein. Möglicherweise hat auch jemand versucht, einen Benutzer mit einem ungültigen Namen hinzuzufügen (früher war es eine beliebte Vorgehensweise, einen Benutzer namens „toor:0:0“ zu erstellen und so Root-Berechtigungen zu erhalten).

Ergibt der Test, dass die Compliance-Anforderungen nicht erfüllt werden, dann muss der Administrator die entsprechenden Zeilen aus der Datei **`/etc/passwd`** korrigieren oder entfernen.

passwd_zero_uid

Syntax

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

Diese integrierte Funktion stellt sicher, dass nur ein Konto in **/etc/passwd** die UID „0“ aufweist. Diese UID sollte zwar eigentlich dem Konto „root“ vorbehalten sein, doch ist es möglich, weitere Konten mit der UID 0 hinzuzufügen, die dann über Zugriff mit den gleichen Berechtigungen verfügen. Dieser Test ist erfolgreich, wenn genau ein Konto die UID 0 hat; andernfalls schlägt er fehl.

Die UID 0 gewährt Root-Berechtigungen auf dem System. Ein Root-Benutzer kann beliebige Handlungen auf dem System ausführen: Er kann z. B. im Speicher anderer Prozesse (oder des Kernels) spionieren, beliebige Dateien auf dem System lesen und schreiben usw. Weil dieses Konto so viele Möglichkeiten hat, muss seine Verwendung auf ein Minimum beschränkt und es optimal geschützt werden.

Empfehlungen für die Administration sehen vor, dass jede UID eindeutig sein muss. Wenn zwei (oder mehr) Konten mit Root-Berechtigungen vorhanden sind, ist die eindeutige Zuständigkeit eines Systemadministrators für das System nicht mehr gegeben. Außerdem beschränken viele Systeme das Root-Konto auf eine direkte Anmeldung über die Konsole, sodass eine administrative Nutzung nachvollzogen werden kann. In der Regel müssen sich Systemadministratoren zunächst unter ihrem eigenen Konto anmelden und verwenden dann den Befehl **su**, um Root zu werden. Diese Beschränkung wird durch eine zusätzliche UID 0 umgangen.

Wenn mehrere Benutzer Root-Zugriff benötigen, verwenden Sie stattdessen ein Tool wie **sudo** oder **calife** (oder RBAC unter Solaris). Es darf nur ein Konto mit der UID 0 geben.

passwd_duplicate_uid

Syntax

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

Diese integrierte Funktion stellt sicher, dass jedes Konto, das in **/etc/passwd** aufgeführt ist, eine eindeutige UID aufweist. Dieser Test ist erfolgreich, wenn alle UIDs eindeutig sind; andernfalls schlägt er fehl.

Jedem Benutzer eines UNIX-Systems ist eine Benutzer-ID (UID) zugeordnet. Hierbei handelt es sich um eine Zahl zwischen 0 und 65.535. Wenn zwei Benutzer dieselbe UID verwenden, haben sie nicht nur dieselben Berechtigungen, sondern das System betrachtet sie als dieselbe Person. Hierdurch wird jede Form der Nachvollziehbarkeit unterlaufen, da es unmöglich ist, zu sagen, welche Aktionen von welchem Benutzer ausgeführt wurden (normalerweise führt das System für Logdateien einen Reverse-Lookup der UID aus und listet dann das erste gefundene Konto mit der betreffenden UID auf).

Sicherheitsstandards wie die CIS-Benchmarks untersagen die Mehrfachverwendung von UIDs für mehrere Benutzer. Wenn Benutzer Dateien gemeinsam verwenden müssen, sollten Sie stattdessen Gruppen anlegen.

Weisen Sie jedem Benutzer auf dem System eine eindeutige ID zu.

passwd_duplicate_uid

Syntax

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

Diese integrierte Funktion stellt sicher, dass die primäre Gruppen-ID (GID) jedes Benutzers eindeutig ist. Dieser Test ist erfolgreich, wenn jeder Benutzer über eine eindeutige GID verfügt; andernfalls schlägt er fehl.

Sicherheitsstandards empfehlen die Erstellung einer Gruppe pro Benutzer (deren Name normalerweise mit dem Benutzernamen identisch ist). Bei einer solchen Konfiguration sind Dateien, die vom Benutzer erstellt werden, normalerweise erst einmal sicher, da sie zu seiner primären Gruppe gehören und deswegen nur vom Benutzer selbst geändert werden können. Möchte der Benutzer, dass auch andere Mitglieder einer Gruppe Besitzer der Datei werden, dann muss er den Besitz mithilfe des Befehls **chgrp** ausdrücklich ändern.

Ein weiterer Vorteil dieser Vorgehensweise besteht darin, dass sie die Verwaltung der Gruppenmitgliedschaft in einer einzigen Datei (**/etc/group**) zusammenfasst, statt sie auf **/etc/passwd** und **/etc/group** zu verteilen.

Erstellen Sie für jeden Benutzer eine Gruppe desselben Namens. Verwalten Sie die Gruppenmitgliedschaft nur über **/etc/group**.

passwd_duplicate_username

Syntax

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

Diese integrierte Funktion stellt sicher, dass jeder Benutzername in **/etc/passwd** eindeutig ist. Ist dies der Fall, dann ist der Test erfolgreich; andernfalls schlägt er fehl.

Doppelt vorhandene Benutzernamen in **/etc/passwd** führen zu Problemen, weil nicht klar ist, welche Berechtigungen angewendet werden.

Mit dem Befehl **adduser** können Sie einen bereits vorhandenen Benutzernamen nicht noch einmal anlegen. Tritt eine solche Konfiguration auf, dann weist dies normalerweise darauf hin, dass das System angegriffen wurde, die für die Benutzerverwaltung verwendeten Tools fehlerhaft sind oder die Datei **/etc/passwd** manuell bearbeitet wurde.

Löschen Sie doppelt vorhandene Benutzernamen, oder ändern Sie sie ab.

passwd_duplicate_home

Syntax

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

Diese integrierte Funktion stellt sicher, dass jeder Nicht-System-Benutzer (d. h. Benutzer mit einer UID, die größer als 100 ist) in **/etc/passwd** ein eindeutiges Benutzerverzeichnis hat.

Jeder Benutzername in **/etc/passwd** benötigt ein eindeutiges Benutzerverzeichnis. Wenn Benutzer ein gemeinsames Verzeichnis verwenden, kann ein Benutzer den oder die anderen Benutzer zur Ausführung beliebiger Befehle zwingen, indem er die Startdateien (**.profile** usw.) modifiziert oder gefälschte Binärdateien im Benutzerverzeichnis ablegt. Außerdem verhindert ein freigegebenes Benutzerverzeichnis die Nachvollziehbarkeit von Benutzeraktionen.

Compliance-Anforderungen sehen vor, dass jeder Benutzer ein eindeutiges Benutzerverzeichnis benötigt.

passwd_shadowed

Syntax

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

Dieser integrierte Test stellt sicher, dass jedes Kennwort in **/etc/passwd** in einer anderen Datei abgelegt ist (Shadow-Kennwort).

Da **/etc/passwd** für alle Benutzer lesbar (world-readable) ist, kann, wenn die Hashes der Benutzerkennwörter darin gespeichert sind, jeder zugriffsberechtigte Benutzer Programme zum Knacken der Kennwörter ausführen. Versuche, das Kennwort eines Benutzers mittels Brute-Force-Attacke (also durch wiederholte Anmeldeversuche, bei denen jedes Mal ein anderes Kennwort ausprobiert wird) zu erraten, lassen sich in den Systemlogdateien normalerweise ausfindig machen. Wenn die Datei **/etc/passwd** die Kennwort-Hashes enthält, könnte sie offline kopiert und dann durch ein Programm zum Knacken von Kennwörtern verarbeitet werden. Auf diese Weise kann ein Angreifer an die Benutzerkennwörter gelangen, ohne entdeckt zu werden.

Die meisten modernen UNIX-Systeme verwenden Shadow-Kennwortdateien. Ziehen Sie Ihre Systemdokumentation zurate, um zu erfahren, wie Sie Shadow-Kennwörter auf Ihrem System aktivieren.

passwd_invalid_gid

Syntax

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

Diese integrierte Funktion stellt sicher, dass jede Gruppen-ID (GID), die in **/etc/passwd** aufgeführt ist, in **/etc/group** vorhanden ist. Wenn alle GIDs korrekt definiert sind, ist der Test erfolgreich; andernfalls schlägt er fehl.

Jedes Mal, wenn eine Gruppen-ID in **/etc/passwd** definiert wird, sollte sie sofort in **/etc/group** aufgeführt werden. Andernfalls ist das System inkonsistent, was zu Problemen führen kann.

Nehmen wir folgenden Fall als Beispiel: Ein Benutzer („bob“) verwendet eine UID von 1000 und GID von 4000. Die GID ist in **/etc/group** nicht definiert, d. h., die primäre Gruppe des Benutzers gewährt ihm gegenwärtig keine Berechtigungen. Wenige Monate später bearbeitet der Systemadministrator **/etc/group**, fügt die Gruppe „admin“ hinzu

und wählt dann die vermeintlich ungenutzte GID #4000 zu deren Identifizierung aus. Von nun an gehört der Benutzer „bob“ standardmäßig zur Gruppe „admin“, obwohl dies gar nicht vorgesehen war.

Bearbeiten Sie **/etc/group**, um die fehlenden GIDs hinzuzufügen.

Gruppendateiverwaltung

group_file_consistency

Syntax

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

Diese integrierte Funktion stellt sicher, dass jede Zeile in **/etc/group** ein gültiges Format aufweist (z. B. drei Felder, die durch Doppelpunkt getrennt sind, und eine Liste mit Benutzern). Ist eine Zeile fehlerhaft formatiert, dann wird dies gemeldet, und der Test schlägt fehl.

Die Verwendung einer fehlformatierten Datei **/etc/group** kann dazu führen, dass verschiedene Tools zur Benutzerverwaltung nicht mehr funktionieren. Weiterhin kann dies ein Hinweis auf ein Eindringen oder einen Bug in einer selbstentwickelten Benutzerverwaltungsanwendung sein. Es kann auch zeigen, dass jemand versucht hat, einen Benutzer mit einem ungültigen Gruppennamen hinzuzufügen.

Bearbeiten Sie die Datei **/etc/group**, um die fehlformatierten Zeilen zu korrigieren.

group_zero_gid

Syntax

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

Diese integrierte Funktion stellt sicher, dass nur eine Gruppe die Gruppen-ID (GID) 0 hat. Dieser Test ist erfolgreich, wenn genau eine Gruppe die GID 0 hat; andernfalls schlägt er fehl.

Die GID 0 bezeichnet eine Gruppe, deren Mitglieder ebenfalls in der primären Gruppe von „root“ enthalten sind. Sie erhalten auf diese Weise Root-Berechtigungen für alle Dateien mit Root-Gruppenberechtigungen.

Wenn Sie eine Gruppe mit Administratoren definieren möchten, erstellen Sie stattdessen eine Gruppe „admin“.

group_duplicate_name

Syntax

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

Dieser integrierte Test stellt sicher, dass jeder Gruppenname eindeutig ist. Ist dies der Fall, dann ist der Test erfolgreich; andernfalls schlägt er fehl.

Doppelt vorhandene Gruppennamen in **`/etc/group`** führen zu Problemen, weil nicht klar ist, welche Berechtigungen angewendet werden. Das bedeutet, dass eine Gruppe mit doppelt vorhandenem Namen Mitglieder und Berechtigungen haben könnte, die ihm überhaupt nicht zustehen.

Löschen Sie doppelt vorhandene Gruppennamen, oder benennen Sie die entsprechenden Gruppen um.

group_duplicate_gid

Syntax

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

Jeder Gruppe in einem UNIX-System ist eine Gruppen-ID (GID) zugeordnet. Hierbei handelt es sich um eine Zahl zwischen 0 und 65.535. Wenn zwei Gruppen dieselbe GID verwenden, haben sie nicht nur dieselben Berechtigungen, sondern das System betrachtet sie als dieselbe Gruppe. Dies unterläuft den Zweck der Verwendung von Gruppen zur Trennung von Benutzerrechten.

Sicherheitsstandards untersagen die gemeinsame Benutzung einer GID durch mehrere Gruppen. Wenn zwei Gruppen dieselben Berechtigungen benötigen, sollten Sie dieselben Benutzer enthalten.

Löschen Sie doppelt vorhandene Gruppen, oder weisen Sie einer dieser Gruppen eine neue eindeutige GID zu.

group_duplicate_members

Syntax

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

Diese integrierte Funktion stellt sicher, dass jedes Mitglied einer Gruppe nur einmal aufgeführt ist. Wenn alle Mitglieder eindeutig sind, ist der Test erfolgreich; andernfalls schlägt er fehl.

Jedes Mitglied einer Gruppe darf nur einmal aufgeführt werden. Zwar stellt die mehrfache Auflistung für das zugrunde liegende Betriebssystem kein Problem dar, aber sie erschwert die Arbeit des Administrators, da insbesondere das Widerrufen von Berechtigungen komplizierter wird. Wenn beispielsweise die Gruppe „admin“ die Mitglieder „alice,bob,charles,daniel,bob“ hat, muss „bob“ zweimal entfernt werden, wenn seine Berechtigungen widerrufen werden sollen.

Stellen Sie sicher, dass jedes Mitglied nur einmal aufgeführt ist.

group_nonexistant_users

Syntax

```
<item>
 name: "group_nonexistant_users"
```

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

Mit diesem Test wird sichergestellt, dass jedes Mitglied einer Gruppe tatsächlich in **/etc/passwd** vorhanden ist.

Sind nicht vorhandene Benutzer in **/etc/group** aufgeführt, so lässt dies auf eine unvollständige Administration schließen. Der Benutzer ist nicht vorhanden, weil entweder die Namenseingabe fehlerhaft war oder der Benutzer beim Entfernen aus dem System nicht aus der Gruppe entfernt wurde.

Es wird empfohlen, solche „Geisterbenutzer“ aus **/etc/group** zu entfernen. Wenn ein Benutzer mit demselben Benutzernamen später hinzugefügt werden sollte, könnte dieser Benutzer Gruppenberechtigungen erhalten, die für ihn nicht vorgesehen sind.

Entfernen Sie nicht vorhandene Benutzer aus **/etc/group**.

Root-Umgebung

dot_in_root_path_variable

Syntax

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

Dieser Test stellt sicher, dass das aktuelle Arbeitsverzeichnis („.“) nicht im Pfad für ausführbare Dateien des Root-Benutzers enthalten ist. Hierdurch wird verhindert, dass ein Angreifer sich Superuser-Berechtigungen verschafft, indem er einen als „root“ angemeldeten Administrator zur Ausführung eines trojanischen Pferdes zwingt, das im aktuellen Arbeitsverzeichnis installiert sein könnte.

writeable_dirs_in_root_path_variable

Syntax

```
<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>
```

Dieser Test meldet alle Verzeichnisse in der PATH-Variable des Root-Benutzers, die Schreibrechte für alle (world-writeable) bzw. für Gruppen (group-writeable) aufweisen. Alle Verzeichnisse, die von diesem Test zurückgegeben werden, sollten sorgfältig überprüft werden, und Schreibberechtigungen für alle Benutzer bzw. für Gruppen sollten erforderlichenfalls wie folgt entfernt werden:

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

Dateiberechtigungen

find_orphan_files

Syntax

```
<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>
```

Dieser Test meldet alle verwaisten Dateien auf dem System.

Standardmäßig wird die Suche rekursiv beginnend beim Verzeichnis „/“ durchgeführt. Dies kann je nach Anzahl der Dateien auf dem Remotesystem bewirken, dass der Test extrem langsam ausgeführt wird. Allerdings kann, falls notwendig, das vorgegebene Ausgangsverzeichnis für die Suche mithilfe des optionalen Schlüsselwortes **basedir** geändert werden. Zudem können bestimmte Dateien in einem Ausgangsverzeichnis auch übersprungen werden; hierzu dient das Schlüsselwort **ignore**. Beim Durchsuchen von Dateisystemen werden standardmäßig alle Verzeichnisse ignoriert, die über NFS angebunden sind; sie lassen sich jedoch über das optionale Schlüsselwort **dir** in die Suche einbeziehen.

Aufgrund der Funktionalität dieses Tests ist je nach Art des gescannten Systems eine Ausführungsdauer von mehreren Stunden normal. Ein Wert von fünf Stunden wurde als Vorgabe für einen Timeout festgelegt, d. h., Ergebnisse dieses Tests werden nach Verstreichen dieses Zeitraums von Nessus nicht mehr verarbeitet. Dieser Wert kann nicht geändert werden.

Beispiel:

```
 <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

Syntax

```
<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>
```

Dieser Test meldet alle Dateien auf dem Remotesystem, in die von allen Benutzern geschrieben werden kann (world-writeable). Im Idealfall sollten auf dem System keine derartigen Dateien vorhanden sein, d. h., das Testergebnis sollte leer sein. In manchen Fällen kann es jedoch abhängig von den geschäftlichen Anforderungen notwendig sein, solche Dateien zu verwenden. Alle Elemente, die von diesem Test zurückgegeben werden, müssen sorgfältig überprüft werden, und Dateien, die Schreibberechtigungen für alle Benutzer aufweisen, sollten erforderlichenfalls wie folgt entfernt werden:

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

Standardmäßig wird die Suche rekursiv beginnend beim Verzeichnis „/“ durchgeführt. Dies kann je nach Anzahl der Dateien auf dem Remotesystem bewirken, dass der Test extrem langsam ausgeführt wird. Allerdings kann, falls notwendig, das vorgegebene Ausgangsverzeichnis für die Suche mithilfe des optionalen Schlüsselwortes **basedir** geändert werden. Zudem können bestimmte Dateien in einem Ausgangsverzeichnis auch übersprungen werden; hierzu dient das Schlüsselwort **ignore**. Beim Durchsuchen von Dateisystemen werden standardmäßig alle Verzeichnisse ignoriert, die über NFS angebunden sind; sie lassen sich jedoch über das optionale Schlüsselwort **dir** in die Suche einbeziehen.

Aufgrund der Funktionalität dieses Tests ist je nach Art des gescannten Systems eine Ausführungsdauer von mehreren Stunden normal. Ein Wert von fünf Stunden wurde als Vorgabe für einen Timeout festgelegt, d. h., Ergebnisse dieses Tests werden nach Verstreichen dieses Zeitraums von Nessus nicht mehr verarbeitet. Dieser Wert kann nicht geändert werden.

Beispiel:

```
<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

Syntax

```
<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>
```

Dieser Test meldet alle Verzeichnisse auf dem Remotesystem, in die von allen Benutzern geschrieben werden kann (world-writeable) und deren Sticky-Bit nicht gesetzt ist. Die Überprüfung, ob das Sticky-Bit für alle World-writeable-Verzeichnisse gesetzt ist, gewährleistet, dass nur der Besitzer einer Datei in einem Verzeichnis diese auch löschen kann. Hierdurch wird verhindert, dass ein anderer Benutzer die Datei versehentlich oder gewollt löscht.

Standardmäßig wird die Suche rekursiv beginnend beim Verzeichnis „/“ durchgeführt. Dies kann je nach Anzahl der Dateien auf dem Remotesystem bewirken, dass der Test extrem langsam ausgeführt wird. Allerdings kann, falls

notwendig, das vorgegebene Ausgangsverzeichnis für die Suche mithilfe des optionalen Schlüsselwortes **basedir** geändert werden. Zudem können bestimmte Dateien in einem Ausgangsverzeichnis auch übersprungen werden; hierzu dient das Schlüsselwort **ignore**. Beim Durchsuchen von Dateisystemen werden standardmäßig alle Verzeichnisse ignoriert, die über NFS angebunden sind; sie lassen sich jedoch über das optionale Schlüsselwort **dir** in die Suche einbeziehen.

Aufgrund der Funktionalität dieses Tests ist je nach Art des gescannten Systems eine Ausführungsdauer von mehreren Stunden normal. Ein Wert von fünf Stunden wurde als Vorgabe für einen Timeout festgelegt, d. h., Ergebnisse dieses Tests werden nach Verstreichen dieses Zeitraums von Nessus nicht mehr verarbeitet. Dieser Wert kann nicht geändert werden.

Beispiel:

```
 <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

Syntax

```
<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>
```

Dieser Test meldet alle Dateien, die für alle Benutzer lesbar („world-readable“) sind. Durch die Überprüfung auf lesbare Dateien beispielsweise in Homeverzeichnissen wird sichergestellt, dass ein Zugriff auf sensible Dateien (z. B. private SSH-Schlüssel) durch Dritte nicht möglich ist.

Standardmäßig wird die Suche rekursiv beginnend beim Verzeichnis „/“ durchgeführt. Dies kann je nach Anzahl der Dateien auf dem Remotesystem bewirken, dass der Test extrem langsam ausgeführt wird. Allerdings kann, falls notwendig, das vorgegebene Ausgangsverzeichnis für die Suche mithilfe des optionalen Schlüsselwortes **basedir** geändert werden. Zudem können bestimmte Dateien in einem Ausgangsverzeichnis auch übersprungen werden; hierzu dient das Schlüsselwort **ignore**. Beim Durchsuchen von Dateisystemen werden standardmäßig alle Verzeichnisse ignoriert, die über NFS angebunden sind; sie lassen sich jedoch über das optionale Schlüsselwort **dir** in die Suche einbeziehen.

Aufgrund der Funktionalität dieses Tests ist je nach Art des gescannten Systems eine Ausführungsdauer von mehreren Stunden normal. Ein Wert von fünf Stunden wurde als Vorgabe für einen Timeout festgelegt, d. h., Ergebnisse dieses Tests werden nach Verstreichen dieses Zeitraums von Nessus nicht mehr verarbeitet. Dieser Wert kann nicht geändert werden.

Beispiel:

```
 <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

Syntax

```
<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>
```

Dieser Test meldet alle Dateien, deren SUID-/SGID-Bit gesetzt ist. Alle von diesem Test gemeldeten Dateien sollten sorgfältig überprüft werden. Dies gilt insbesondere für Shell-Skripts und eigenentwickelte ausführbare Dateien, also etwa solche, die nicht mit dem System ausgeliefert wurden. SUID-/SGID-Dateien bergen das Risiko der Eskalation von Berechtigungen eines normalen Benutzers auf die des Besitzers oder der Gruppe der Datei. Wenn solche Dateien oder Skripte vorhanden sein müssen, sollten sie sorgfältig darauf geprüft werden, ob sie das Erstellen von Dateien mit hochgestuften Berechtigungen gestatten.

Standardmäßig wird die Suche rekursiv beginnend beim Verzeichnis „/“ durchgeführt. Dies kann je nach Anzahl der Dateien auf dem Remotesystem bewirken, dass der Test extrem langsam ausgeführt wird. Allerdings kann, falls notwendig, das vorgegebene Ausgangsverzeichnis für die Suche mithilfe des optionalen Schlüsselwortes **basedir** geändert werden. Zudem können bestimmte Dateien in einem Ausgangsverzeichnis auch übersprungen werden; hierzu dient das Schlüsselwort **ignore**. Beim Durchsuchen von Dateisystemen werden standardmäßig alle Verzeichnisse ignoriert, die über NFS angebunden sind; sie lassen sich jedoch über das optionale Schlüsselwort **dir** in die Suche einbeziehen.

Aufgrund der Funktionalität dieses Tests ist je nach Art des gescannten Systems eine Ausführungsdauer von mehreren Stunden normal. Ein Wert von fünf Stunden wurde als Vorgabe für einen Timeout festgelegt, d. h., Ergebnisse dieses Tests werden nach Verstreichen dieses Zeitraums von Nessus nicht mehr verarbeitet. Dieser Wert kann nicht geändert werden.

Beispiel:

```
 <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
Mit diesem integrierten Test wird überprüft, ob Besitzer einer Lokalisierungsdatei im Homeverzeichnis eines Benutzers der Benutzer selbst oder „root“ ist.

Mit dem Token „file“ können eine oder mehrere Dateien aufgelistet werden. Ist das Token „file“ jedoch nicht vorhanden, dann wird im Verlauf dieses Tests standardmäßig nach den folgenden Dateien gesucht:

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

Beispiel:

```
<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
Mit diesem integrierten Test wird überprüft, ob Besitzer einer Lokalisierungsdatei im Homeverzeichnis eines Benutzers die primäre Gruppe des Benutzers oder „root“ ist.

Mit dem Token „file“ können eine oder mehrere Dateien aufgelistet werden. Ist das Token „file“ jedoch nicht vorhanden, dann wird im Verlauf dieses Tests standardmäßig nach den folgenden Dateien gesucht:

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

Beispiel:

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

Verdächtige Dateiinhalte
admin_accounts_in_ftpusers

Syntax

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

Mit diesem Test wird geprüft, ob alle Administratorkonten – also alle Benutzer mit einer UID kleiner als 500 – in **`/etc/ftpusers`**, **`/etc/ftpd/ftpusers`** oder **`/etc/vsftpd.ftpusers`** vorhanden sind.

Nicht benötigte Dateien
find_pre-CIS_files

Syntax

```
<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>
```

Dieser Test ist für eine bestimmte CIS-Anforderung (Center for Internet Security) maßgeschneidert, um die Zertifizierung für das CIS-Benchmark für Red Hat zu erhalten. Der Test ist besonders nützlich für Anwender, die ein Red Hat-System auf der Basis dieses Benchmarks konfiguriert bzw. gehärtet haben. Das CIS-Benchmark-Tool stellt ein Skript bereit, mit dem alle Systemdateien, die während des Systemhärtungsprozesses geändert wurden, gesichert und mit dem Schlüsselwort „**`-preCIS`**“ versehen werden. Diese Dateien müssen entfernt werden, sobald alle Benchmark-Empfehlungen erfolgreich angewendet wurden und der Betriebszustand des Systems wiederhergestellt wurde. Mit diesem Test wird sichergestellt, dass keine **`preCIS`**-Dateien mehr auf dem Remotesystem vorhanden sind.

Standardmäßig wird die Suche rekursiv beginnend beim Verzeichnis „/“ durchgeführt. Dies kann je nach Anzahl der Dateien auf dem Remotesystem bewirken, dass der Test extrem langsam ausgeführt wird. Allerdings kann, falls notwendig, das vorgegebene Ausgangsverzeichnis für die Suche mithilfe des optionalen Schlüsselwortes **`basedir`** geändert werden. Zudem können bestimmte Dateien in einem Ausgangsverzeichnis auch übersprungen werden; hierzu dient das Schlüsselwort **`ignore`**.

Aufgrund der Funktionalität dieses Tests ist je nach Art des gescannten Systems eine Ausführungsdauer von mehreren Stunden normal. Ein Wert von fünf Stunden wurde als Vorgabe für einen Timeout festgelegt, d. h., Ergebnisse dieses Tests werden nach Verstreichen dieses Zeitraums von Nessus nicht mehr verarbeitet. Dieser Wert kann nicht geändert werden.

Bedingungen

Sie können eine **if**/**then**/**else**-Logik in der UNIX-Richtlinie definieren. Auf diese Weise muss der Endbenutzer nur eine einzige Datei verwenden, die mehrere Konfigurationen verarbeiten kann. Beispielsweise kann dieselbe Richtliniendatei die Einstellungen sowohl für Postfix als auch für Sendmail testen, wenn die korrekte **if**/**then**/**else**-Syntax eingesetzt wird.

Die Syntax zur Ausführung von Bedingungen lautet wie folgt:

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

Beispiel:

```
<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>
```

Ob die Bedingung erfüllt wird oder nicht, wird im Bericht nicht angezeigt, da es sich hier um einen im Hintergrund stattfindenden Test handelt.

Bei den Bedingungen wird zwischen den Typen „**and**“ und „**or**“ unterschieden.

NetApp Data ONTAP

In diesem Abschnitt werden Format und Funktionen von Speichersystemen, auf denen NetApp Data ONTAP-Compliancetests laufen, und das Prinzip hinter den einzelnen Einstellungen erläutert.

Verwendung von Anführungszeichen:

Um Auditfelder gesetzte einzelne bzw. doppelte Anführungszeichen werden austauschbar verwendet. Ausgenommen sind lediglich die folgenden Fälle:

1. Bei Windows-Compliancetests, in denen Sonderfelder (z. B. CRLF) literal interpretiert werden müssen, müssen einzelne Anführungszeichen gesetzt werden. Eingebettete Felder, die als Strings zu interpretieren sind, müssen mit Escapezeichen versehen werden.

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

2. Doppelte Anführungszeichen sind erforderlich, wenn „include_paths“ und „exclude_paths“ für WindowsFiles verwendet werden.

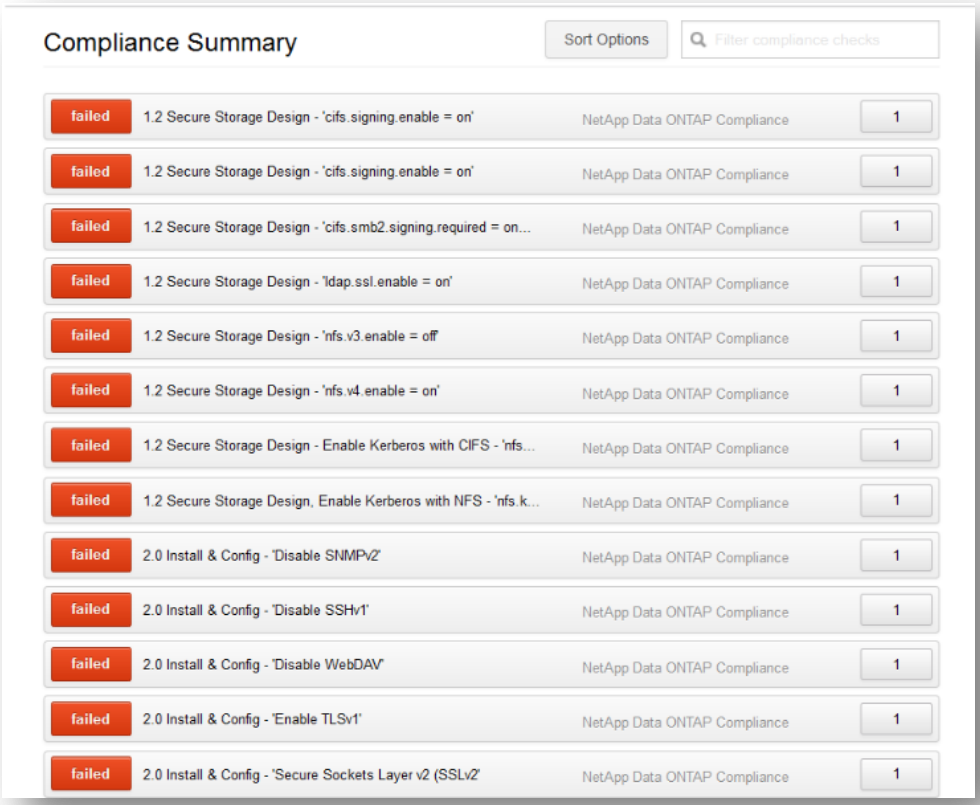
Wenn Strings in einem beliebigen Feldtyp (`description`, `value_data`, `regex` usw.) verwendet werden, die doppelte Anführungszeichen aufweisen, kann auf zweierlei Weise verfahren werden:

a. Verwenden Sie den jeweils anderen Anführungszeichentyp als äußere umschließende Anführungszeichen.

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

b. Stellen Sie eingebetteten doppelten Anführungszeichen den umgekehrten Schrägstrich („\“) als Escapezeichen voran.

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



Erforderliche Benutzerberechtigungen

Zur Durchführung eines erfolgreichen Compliancescans für ein NetApp Data ONTAP-System sind für authentifizierte Benutzer die folgenden Berechtigungen erforderlich:

`root`-Anmeldedaten für NetApp Data ONTAP-Filter

Neben den oben genannten Berechtigungen werden eine Auditrichtlinie für NetApp Data ONTAP-Compliancetests und das Nessus-Plugin mit der ID 66934 („NetApp Data ONTAP Compliance Checks“) benötigt.

Zum Scannen des Geräts richten Sie zunächst eine Auditrichtlinie ein. Verwenden Sie dann das Menü „**SSH settings**“ („SSH-Einstellungen“) auf der Registerkarte „**Credentials**“ („Anmeldedaten“) der Richtlinie, um die `root`-Anmeldedaten einzugeben. Wählen Sie auf der Registerkarte „**Plugins**“ die Plugin-Familie „Policy Compliance“ („Richtliniencompliance“) und aktivieren Sie das Plugin mit der ID 66934 („**NetApp Data ONTAP Compliance Checks**“). Wählen Sie dann auf der Registerkarte „**Preferences**“ („Voreinstellungen“) das Dropdownmenü „NetApp Data ONTAP Compliance Checks“ („NetApp Data ONTAP-Compliancetests“)aus und fügen Sie die NetApp-Auditdatei aus dem Tenable Support Portal hinzu. Speichern Sie dann die Richtlinie und führen Sie den Scan aus.

In Fällen, in denen die Angabe von `root`-Berechtigungen nicht zur Wahl steht, kann ein Konto mit niedrigeren Berechtigungen erstellt werden, um das Audit zu ermöglichen:

1. Erstellen Sie eine neue Rolle (z. B. `nessus_audit`):

   ```
   # role add nessus_audit -a login-ssh,cli-version,cli-options,cli-uptime
   ```

2. Weisen Sie die Rolle einer Gruppe zu (z. B. `nessus_admins`):

   ```
   # group add nessus_admins -r nessus_audit
   ```

3. Weisen Sie der Gruppe einen Benutzer zu:

```
# useradmin user add nessus -g nessus_admins
```

Testtyp: CONFIG_CHECK

NetApp Data ONTAP-Compliancetests sind in **custom_item**- Elementen gekapselt und haben den Typ CONFIG_CHECK. Sie werden wie alle anderen **.audit**-Dateien verarbeitet und funktionieren auf Systemen mit dem NetApp Data ONTAP-System. Der „CONFIG_CHECK“-Test umfasst mindestens zwei Schlüsselwörter. Die Schlüsselwörter **type** und **description** sind Pflichtangaben, auf die mindestens ein weiteres Schlüsselwort folgt. Der Test funktioniert durch Prüfung der Ausgabe des Befehls „options“.

Schlüsselwörter

Die folgende Tabelle gibt an, wie die einzelnen Schlüsselwörter in NetApp Data ONTAP-Compliancetests verwendet werden können:

Schlüsselwort	Einsatzbeispiel und unterstützte Einstellungen
type	„CHECK_CONFIG“ bestimmt, ob das angegebene Konfigurationselement in der NetApp Data ONTAP-Ausgabe von „show configuration“ vorhanden ist.
description	Das Schlüsselwort „**description**“ bietet die Möglichkeit, dem ausgeführten Test eine kurze Beschreibung hinzuzufügen. Es wird dringend empfohlen, dem Feld einen eindeutigen Wert zuzuweisen und sicherzustellen, dass andere Tests keinen identischen Eintrag in diesem Feld aufweisen. Auf der Basis des Feldes generiert Tenable SecurityCenter automatisch eine eindeutige Plugin-ID. Beispiel: `description: "1.0 Require strong Password Controls - 'min-password-length >= 8'"`
info	Das Schlüsselwort „**info**“ wird verwendet, um dem ausgeführten Test eine ausführlichere Beschreibung hinzuzufügen. Dies könnte ein Verweis auf eine Rechtsvorschrift, eine URL mit weiteren Informationen, eine Unternehmensrichtlinie usw. sein. Mehrere **info**-Felder lassen sich in separaten Zeilen hinzufügen, um den Text als Abschnitt zu formatieren. Die Anzahl der verwendbaren **info**-Felder ist nicht begrenzt.  Jedes „**info**“-Tag muss in eine einzelne Zeile ohne Zeilenumbruch geschrieben werden. Wenn mehrere Zeilen erforderlich sind (z. B. zu Formatierungszwecken), fügen Sie einfach weitere „**info**“-Tags hinzu. Beispiel: `info: "Enable palindrome-check on passwords"`
severity	Das Schlüsselwort „**severity**“ gibt die Intensität (den Schweregrad) des ausgeführten Tests an. Beispiel: `severity: MEDIUM` Die Intensität kann auf HIGH („Hoch“), MEDIUM („Moderat“) oder LOW („Niedrig“) festgelegt werden.
regex	Mit dem Schlüsselwort „**regex**“ aktivieren Sie die Suche nach der Konfigurationselementeinstellung, die einem bestimmten regulären Ausdruck

	entspricht. Beispiel: `regex: "set snmp .+"` Die folgenden Metazeichen erfordern einen gesonderten Umgang: + \ * ( ) ^ Wenn diese Zeichen literal ausgewertet werden sollen, müssen Sie sie entweder mit zwei umgekehrten Schrägstrichen („\\“) als Escapezeichen auszeichnen oder sie in eckige Klammern („[]“) setzen. Die folgenden Zeichen hingegen müssen für eine literale Auswertung mit nur einem umgekehrten Schrägstrich ausgezeichnet werden: . ? " ' . Dies hat mit der Art und Weise zu tun, wie der Compiler diese Zeichen auswertet. Wenn das Tag „`regex`“ für einen Test festgelegt, aber keines der Tags „expect“, „not_expect“ oder „number_of_lines“ angegeben wurde, meldet der Test alle Zeilen mit diesem regulären Ausdruck.

`expect`	Das Schlüsselwort ermöglicht die Prüfung des Konfigurationselements mit dem passenden „**`regex`**“-Tag. Wurde kein „**`regex`**“-Tag verwendet, sucht der Test in der gesamten Konfiguration nach dem „**`expect`**“-String.
	
	Der Test ist bestanden, sofern die von „**`regex`**“ gefundene Konfigurationszeile dem Tag „**`expect`**“ entspricht oder, wenn „**`regex`**“ nicht verwendet wurde, der „**`expect`**“-String in der Konfiguration gefunden wurde.
	
	Beispiel:
	`regex: "set password-controls complexity"`
	`expect: "set password-controls complexity [1-4]"`
	
	Im oben stehenden Beispiel stellt das Tag „**`expect`**“ sicher, dass die Komplexität auf einen Wert zwischen 1 und 4 festgelegt ist.

`not_expect`	Dieses Schlüsselwort ermöglicht die Suche nach Konfigurationselementen, die nicht in der Konfiguration vorhanden sein sollten.
	
	Es bewirkt das Gegenteil von „**`expect`**“. Der Test ist bestanden, sofern die von „**`regex`**“ gefundene Konfigurationszeile dem Tag „**`not_expect`**“ nicht entspricht oder, wenn „**`regex`**“ nicht verwendet wurde, der „**`not expect`**“-String in der Konfiguration nicht gefunden wurde.
	
	Beispiel:
	`regex: "set password-controls password-expiration"`
	`not_expect: "set password-controls password-expiration never"`
	
	Im oben stehenden Beispiel stellt das Tag „**`not_expect`**“ sicher, dass „password-controls“ nicht auf „never“ festgelegt ist.

CONFIG_CHECK Beispiele

Folgende Beispiele zeigen die Verwendung von CONFIG_CHECK auf einem NetApp Data ONTAP-Gerät:

```
<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>
```

Bedingungen

Sie können eine **`if`**/**`then`**/**`else`**-Logik in der NetApp Data ONTAP -Auditrichtlinie definieren. Auf diese Weise muss der Endbenutzer nur eine einzige Datei verwenden, die mehrere Konfigurationen verarbeiten kann.

Die Syntax zur Ausführung von Bedingungen lautet wie folgt:

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

Beispiel:

```
<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>
```

Die Bedingung taucht nie im Bericht auf, d. h. es handelt sich um einen im Hintergrund stattfindenden Test, bei dem Erfolg oder Fehlschlagen nicht gemeldet werden.

Bei den Bedingungen wird zwischen den Typen „**and**“ und „**or**“ unterschieden.

Berichterstellung

Kann in einem <then>- oder <else>-Abschnitt durchgeführt werden, um die gewünschte PASSED/FAILED-Bedingung zu erhalten.

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

PASSED, WARNING und FAILED sind zulässige Werte für „report type“.

Referenz zu Compliancedateien für IBM iSeries-Konfigurationsaudits

In diesem Abschnitt werden Format und Funktionen von IBM iSeries-Compliancetests und das Prinzip hinter den einzelnen Einstellungen erläutert.

Verwendung von Anführungszeichen:

Um Auditfelder gesetzte einzelne bzw. doppelte Anführungszeichen werden austauschbar verwendet. Ausgenommen sind lediglich die folgenden Fälle:

1. Bei Windows-Compliancetests, in denen Sonderfelder (z. B. CRLF) literal interpretiert werden müssen, müssen einzelne Anführungszeichen gesetzt werden. Eingebettete Felder, die als Strings zu interpretieren sind, müssen mit Escapezeichen versehen werden.

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

2. Doppelte Anführungszeichen sind erforderlich, wenn „include_paths“ und „exclude_paths“ für WindowsFiles

verwendet werden.

Wenn Strings in einem beliebigen Feldtyp (`description`, `value_data`, `regex` usw.) verwendet werden, die doppelte Anführungszeichen aufweisen, kann auf zweierlei Weise verfahren werden:

a. Verwenden Sie den jeweils anderen Anführungszeichentyp als äußere umschließende Anführungszeichen.

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

b. Stellen Sie eingebetteten doppelten Anführungszeichen den umgekehrten Schrägstrich („\“) als Escapezeichen voran.

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

Erforderliche Benutzerberechtigungen

Zur Durchführung eines erfolgreichen Compliancescans für ein iSeries-System sind für authentifizierte Benutzer die folgenden Berechtigungen erforderlich:

1. Ein Benutzer mit der Berechtigung „(*ALLOBJ)“ oder „audit (*AUDIT)“ kann ein Audit aller Systemwerte durchführen. Solche Benutzer gehören normalerweise zur Klasse „(*SECOFR)“.

2. Benutzer der Klassen „(*USER)“ und „(*SYSOPR)“ können Audits der meisten Werte mit Ausnahme von QAUDCTL, QAUDENDACN, QAUDFRCLVL, QAUDLVL, QAUDLVL2 und QCRTOBJAUD durchführen.

Wenn ein Benutzer nicht über die Berechtigungen verfügt, um auf einen Wert zuzugreifen, wird der Wert „*NOTAVL“ zurückgegeben.

Testtyp

Alle IBM iSeries-Compliancetests müssen in die Kapselung **`check_type`** gesetzt werden, die mit der Typangabe „AS/400“ zu versehen ist. Dies ist erforderlich, um **`.audit`**-Dateien für Systeme, die unter IBM iSeries laufen, von anderen Compliancetests unterscheiden zu können.

Beispiel:

```
<check_type:"AS/400">
```

Anders als bei anderen Compliance-Audittypen sind keine weiteren Schlüsselwörter zur Angabe von Typ oder Version vorhanden.

Schlüsselwörter

Die folgende Tabelle gibt an, wie die einzelnen Schlüsselwörter in IBM iSeries-Compliancetests verwendet werden können:

Schlüsselwort	Einsatzbeispiel und unterstützte Einstellungen
`type`	`AUDIT_SYSTEMVAL` `SHOW_SYSTEMVAL`
`systemvalue`	Mit diesem Schlüsselwort wird ein bestimmter Wert angegeben, der auf dem IBM iSeries-System überprüft werden soll. Beispiel:

	`systemvalue: "QALWUSRDMN"`
description	Dieses Schlüsselwort bietet die Möglichkeit, dem ausgeführten Test eine kurze Beschreibung hinzuzufügen. Es wird dringend empfohlen, dem Feld einen eindeutigen Wert zuzuweisen und sicherzustellen, dass andere Tests keinen identischen Eintrag in diesem Feld aufweisen. Auf der Basis des Feldes **description** generiert Tenable SecurityCenter automatisch eine eindeutige Plugin-ID. Beispiel: `description: "Allow User Domain Objects (QALWUSRDMN) - '*all'"`

value_type	Mit diesem Schlüsselwort wird der Typ („POLICY_DWORD“ oder „POLICY_TEXT“) des Wertes angegeben, der auf dem IBM iSeries-System überprüft wird.
	
	Beispiel:
	`value_type: "POLICY_DWORD"`
	
	Beispiel:
	`value_type: "POLICY_TEXT"`

value_data	Mit diesem Schlüsselwort wird der Datenwert definiert, der für einen Systemwert erwartet wird. Beispiel: `value_type: "^([6-9]|[1-9][0-9]+)$"`

check_type	Mit diesem Schlüsselwort wird der Typ des Tests definiert, mit dem ein Datenwert geprüft wird.
	
	Beispiele:
	`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"`
	
	Beispiel:
	`<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	Dieses Schlüsselwort wird verwendet, um dem ausgeführten Test eine ausführlichere Beschreibung hinzuzufügen. Dies könnte eine Rechtsvorschrift, eine URL, eine Unternehmensrichtlinie oder eine Angabe des Grundes sein, warum diese Einstellung erforderlich ist. Mehrere **info**-Felder lassen sich in separaten Zeilen hinzufügen, um den Text als Abschnitt zu formatieren. Die Anzahl der verwendbaren **info**-Felder ist nicht begrenzt. Beispiel:

```
info: "\nref : http://publib.boulder.ibm.com/infocenter/
iseries/v5r4/topic/books/sc415302.pdf pg. 21"
```

Benutzerdefinierte Elemente

Ein benutzerdefiniertes Element ist ein vollständiger Test, der auf der Basis der oben definierten Schlüsselwörter definiert wird. Die folgende Liste enthält die verfügbaren Typen für benutzerdefinierte Elemente. Jeder Test beginnt mit dem Tag „**<custom_item>**“ und endet mit „**</custom_item>**“. In die Tags eingeschlossen ist eine Liste mit einem oder mehreren Schlüsselwörtern, die vom Compliancetest-Parser zur Durchführung des Tests interpretiert werden.

Bei den benutzerdefinierten Audittests können „**</custom_item>**“ und „**</item>**“ als schließende Tags beliebig gegeneinander ausgetauscht werden.

AUDIT_SYSTEMVAL

Mit „AUDIT_SYSTEMVALUE“ wird der Wert der mit dem Schlüsselwort „**systemvalue**“ angegebenen Konfigurationseinstellung geprüft. Das Schlüsselwort „**check_type**“ gibt dabei an, auf welche Weise der Wert verglichen wird.

```
<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

Mit dem Audit „SHOW_SYSTEMVAL“ wird der Wert der mit dem Schlüsselwort „**systemvalue**“ angegebenen Konfigurationseinstellung lediglich gemeldet.

```
<custom item>
 type: SHOW_SYSTEMVAL
 systemvalue: "QAUDCTL"
 description: "show QAUDCTL value"
 severity: MEDIUM
</custom_item>
```

Bedingungen

Sie können eine **if/then/else**-Logik in der IBM iSeries-Richtlinie definieren. Dies ermöglicht es, dem Endbenutzer statt des Ergebnisses („Passed“/„Failed“) eine Warnung anzuzeigen, falls das Audit erfolgreich ist.

Die Syntax zur Ausführung von Bedingungen lautet wie folgt:

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

```
  <Insert your audit here>
 </then>
 <else>
  <Insert your audit here>
 </else>
</if>
```

Beispiel:

```
<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>
```

Ob die Bedingung erfüllt wird oder nicht, wird im Bericht nicht angezeigt, da es sich hier um einen im Hintergrund stattfindenden Test handelt.

Bei den Bedingungen wird zwischen den Typen „**and**“ und „**or**“ unterschieden.

Referenz zu Compliancedateien für VMware vCenter/ESXi-Konfigurationsaudits

In diesem Abschnitt werden Format und Funktionen von VMware vCenter- und ESXi-Compliancetests und das Prinzip hinter den einzelnen Einstellungen erläutert.

Nessus kann VMware über die nativen APIs testen, indem es die Konfiguration extrahiert und dann das Audit anhand der in der dazugehörigen **`.audit`**-Datei aufgelisteten Tests ausführt.

Anforderungen

Für einen erfolgreichen Compliancescan von VMware-Systemen wird Folgendes benötigt:

1. Administrative Zugriffsrechte auf VMware vCenter oder ESXi. (Tenable hat sowohl für ESXi (die kostenlose Oberfläche zur Verwaltung von VMs auf ESX/ESXi) als auch für vCenter (ein bei VMware zu beziehendes kostenpflichtiges Modul zur Verwaltung mindestens eines ESX/ESXi-Servers) APIs entwickelt. Dieses Plugin kann die Aufgabe entweder mit ESXi- oder vCenter-Anmeldedaten ausführen.)

2. Auditrichtlinie für VMware vCenter/ESXi-Compliancetests.

3. Plugin mit der ID 64455 („VMware vCenter/ESXi Compliance Checks“)

Unterstützte Versionen

Derzeit kann Nessus ESXi 4.x und 5.x sowie vCenter 4.x und 5 testen.

Testtyp

Die Syntax für die VMware-Auditfunktion ist zur Durchführung umfassend auf XPATH und XSL-Transformationen angewiesen.

Das VMware-Audit unterstützt drei Testtypen:

AUDIT_VM

Dieser Testtyp ermöglicht den Test der Einstellungen virtueller Maschinen (weitere Informationen finden Sie in Anhang C):

```
<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

Dieser Testtyp ermöglicht den Test von ESX/ESXi-Servereinstellungen:

```
<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
```

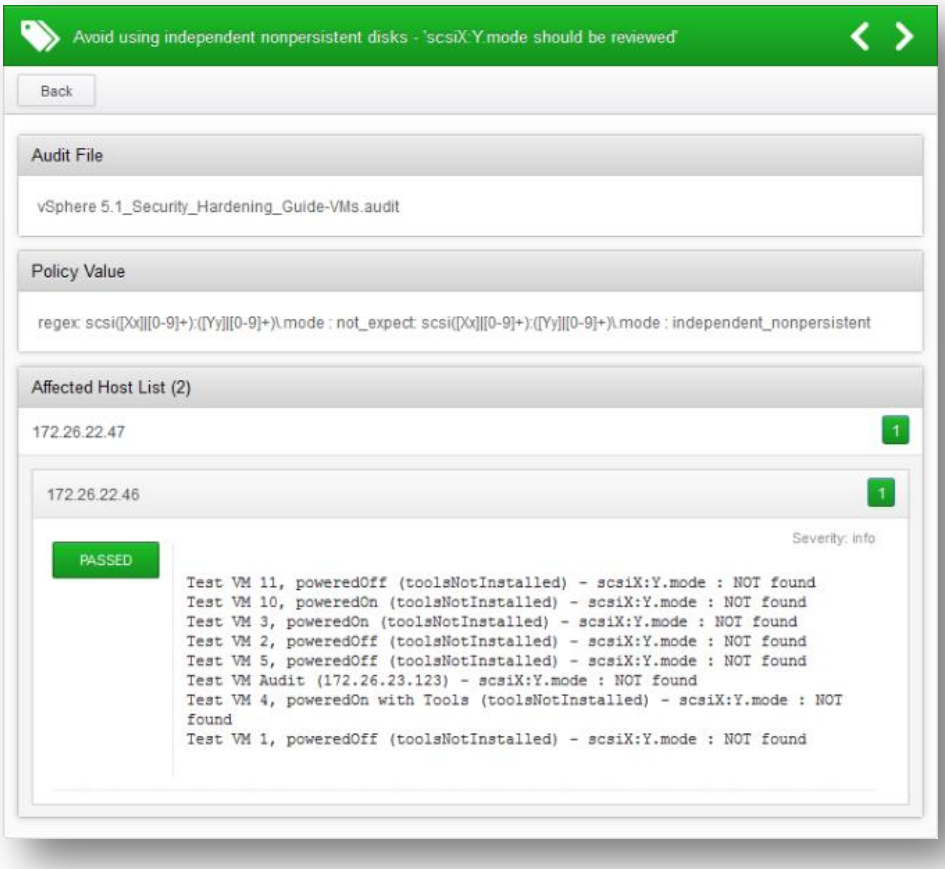
```
      select=\"audit:propSet/audit:val[@xsi:type='HostConfigInfo']/audit:option[audit
      :key[text()='Syslog.global.logDir']]/audit:value\"/>"
 xsl_stmt: "</xsl:template>"
 expect: "Syslog.global.logDir : /foo/bar"
</custom_item>
```

AUDIT_VCENTER

Dieser Testtyp ermöglicht den Test von vCenter-Einstellungen:

```
<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>
```

Beispiel eines bestandenen vSphere-Audits:



Schlüsselwörter

Die folgende Tabelle gibt an, wie die einzelnen Schlüsselwörter in VMware-Compliancetests verwendet werden können:

Schlüsselwort	Einsatzbeispiel und unterstützte Einstellungen
type	Dieses Schlüsselwort beschreibt den Testtyp, der vom jeweiligen Test in einer Auditdatei ausgeführt wird. VMware-Audits verwenden die folgenden drei Audittesttypen: • AUDIT_VM • AUDIT_ESX • AUDIT_VCENTER
description	Dieses Schlüsselwort bietet die Möglichkeit, dem ausgeführten Test eine kurze Beschreibung hinzuzufügen. Es wird empfohlen, dem Feld einen eindeutigen Wert zuzuweisen und sicherzustellen, dass andere Tests keinen identischen Eintrag in diesem Feld aufweisen. Dies ist erforderlich, weil SecurityCenter auf der Basis des Feldes **description** automatisch eine Plugin-ID generiert. Beispiel: `description: "Disconnect unauthorized devices - 'floppyX.present = false'"`
info	Das Schlüsselwort ermöglicht es Benutzern, dem ausgeführten Test eine ausführlichere Beschreibung hinzuzufügen. Es sind mehrere **info**-Felder ohne vordefinierte Einschränkung zulässig. Der Inhalt eines `info`-Feldes sollte in doppelte Anführungszeichen gesetzt werden. Beispiel: `info: "Make sure floppy drive is not attached"`
regex	Mit diesem Schlüsselwort aktivieren Sie die Suche nach Elementen, die einem bestimmten regulären Ausdruck entsprechen. Beispiel: `regex: "floppy([Xx]|[0-9]+)\\.present :"`

Beachten Sie, dass die Compliance eines Tests durch Vergleich der Testausgabe mit einem der Tags „**expect**“ oder „**not_expect**“ bestimmt werden kann. In jedem Test kann nur ein Compliancetest-Tag eingesetzt werden.

Schlüsselwort	Einsatzbeispiel und unterstützte Einstellungen
expect	Das Schlüsselwort ermöglicht die Prüfung des Konfigurationselements mit dem passenden „**regex**“-Schlüsselwort. Wurde kein „**regex**“-Schlüsselwort verwendet, sucht der Test in der gesamten Konfiguration nach dem „**expect**“-String. Der Test ist bestanden, sofern die von „**regex**“ gefundene Konfigurationszeile dem „**expect**“-String entspricht oder, wenn „**regex**“ nicht verwendet wurde, der „**expect**“-String in der Konfiguration gefunden wurde. Beispiel: `regex: "floppy([Xx]|[0-9]+)\\.present :"`

	`expect: floppy([Xx]|[0-9]+)\\.present : false"` Oder: `expect: floppy([Xx]|[0-9]+)\\.present : false"` In den oben stehenden Fällen gewährleistet das Schlüsselwort „**expect**“, dass das Diskettenlaufwerk nicht vorhanden ist.

not_expect	Dieses Schlüsselwort ermöglicht die Suche nach Konfigurationselementen, die nicht in der Konfiguration vorhanden sein sollten.
	
	Es bewirkt das Gegenteil von „**expect**“. Der Test ist bestanden, sofern die von „**regex**“ gefundene Konfigurationszeile dem „**not_expect**“-String-String nicht entspricht oder, wenn das Schlüsselwort „**regex**“ nicht verwendet wurde, der „**not_expect**“-String in der Konfiguration nicht gefunden wurde.
	
	Beispiel:
	
	`regex: floppy([Xx]|[0-9]+)\\.present : "`
	`not_expect: floppy([Xx]|[0-9]+)\\.present : false"`
	
	Oder:
	
	`not_expect: floppy([Xx]|[0-9]+)\\.present : false"`
	
	In den oben stehenden Fällen gewährleistet das Schlüsselwort **expect**, dass das Diskettenlaufwerk nicht vorhanden ist.

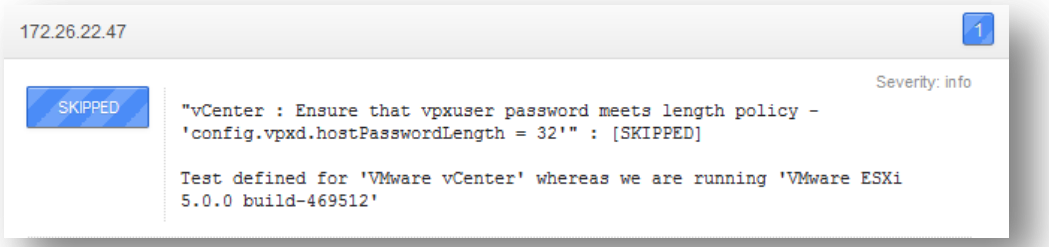
Zusätzliche Hinweise

Wird der Test bestanden, meldet das Plugin alle VMs, die mit dieser Richtlinie übereinstimmen. Das Audit von Tenable meldet sowohl den VM-Namen als auch die IP-Adresse des Ziels. Beachten Sie jedoch, dass die IP-Adresse einer VM nur verfügbar ist, wenn VMware-Tools installiert wurden.

Die Berichte werden wie folgt angezeigt:

```
Test VM 2, poweredOff (toolsNotInstalled) - vmsafe.enable : NOT found
Test VM Audit (172.26.23.123) - vmsafe.enable : NOT found
```

ESX/ESXi und vCenter können mit derselben Richtlinie gescannt werden. Beachten Sie jedoch, dass vCenter-Tests auf ESX/ESXi-Hosts übersprungen werden.



Weitere Informationen

Tenable hat eine Reihe von Dokumenten erstellt, in denen die Bereitstellung, die Installation, die Konfiguration, der Betrieb und die Testmethoden von Nessus ausführlich beschrieben werden.

- **Nessus 5.2 Installation and Configuration Guide** („Nessus 5.2-Installations- und Konfigurationshandbuch“; Schrittanleitung zur Nessus-Installation und -Konfiguration)
- **Nessus 5.2 User Guide** („Nessus 5.2-Benutzerhandbuch“; beschreibt den Einsatz des Nessus Nessus-Sicherheitslückenscanners einschließlich Konfiguration und Berichterstellung)
- **Nessus Credential Checks for Unix and Windows** („Authentifizierte Nessus-Tests für UNIX und Windows“; enthält Informationen zur Durchführung authentifizierter Netzwerkscans mit dem Nessus-Sicherheitslückenscanner)
- **Nessus Compliance Checks** („Nessus-Compliancetests“; allgemeiner Leitfaden zum Verständnis und zur Durchführung von Compliancetests mithilfe von Nessus und SecurityCenter)
- **Nessus v2 File Format** („Nessus V2-Dateiformat“; beschreibt die Struktur des .nessus-Dateiformats, das mit Nessus 3.2 und NessusClient 3.2 eingeführt wurde)
- **Nessus 5.0 REST Protocol Specification** („Nessus 5.0 REST-Protokollspezifikation“; beschreibt das REST-Protokoll und die Schnittstelle in Nessus)
- **Nessus 5 and Antivirus** („Nessus 5 und Virenschutz“; beschreibt die Funktion verschiedener gängiger Sicherheitssoftwarepakete in Nessus und enthält Tipps und Lösungsvorschläge für eine verbesserte Funktionsweise der Software ohne Einschränkung der Sicherheit oder Verhinderung Ihrer Sicherheitslückenscans)
- **Nessus 5 and Mobile Device Scanning** („Nessus 5 und Scans von Mobilgeräten“; beschreibt die Integration von Nessus in Microsoft Active Directory und Verwaltungsserver für Mobilgeräte zur Bestimmung von im Netzwerk eingesetzten Mobilgeräten)
- **Nessus 5.0 and Scanning Virtual Machines** („Nessus 5.0 und Scans virtueller Maschinen“; beschreibt den Einsatz des Sicherheitslückenscanners von Tenable Network Security Nessus für Audits der Konfiguration virtueller Plattformen sowie der darauf ausgeführten Software)
- **Strategic Anti-malware Monitoring with Nessus, PVS, and LCE** („Strategische Malwareüberwachung mit Nessus, PVS und LCE“; beschreibt, wie mithilfe der Tenable USM-Plattform zahlreiche bösartige Softwareprogramme erkannt werden können und das Ausmaß der Malware-Infizierung bestimmt werden kann)

- **Patch Management Integration** („Integration des Patchmanagements“; beschreibt, wie Nessus und SecurityCenter mithilfe von Berechtigungen auf die IBM TEM-, Microsoft WSUS- und SCCM-, VMware Go- und Red Hat Network Satellite-Patchmanagementsysteme Patch-Audits auf Systemen ausführen, für die dem Nessus-Scanner möglicherweise keine Berechtigungen zur Verfügung stehen)
- **Real-Time Compliance Monitoring** („Compliance-Überwachung in Echtzeit“; erläutert, wie die Lösungen von Tenable Sie bei der Erfüllung zahlreicher gesetzlicher Vorschriften und Finanzstandards unterstützen)
- **Tenable Products Plugin Families** („Tenable Produkt-Plugin-Familien“; stellt eine Beschreibung und Zusammenfassung der Plugin-Serien für Nessus, Log Correlation Engine und den Passive Vulnerability Scanner bereit)
- **SecurityCenter Administration Guide** („SecurityCenter-Administratorhandbuch“)

Weitere Onlineressourcen sind nachfolgend aufgeführt:

- Nessus-Diskussionsforum: https://discussions.nessus.org/
- Tenable-Blog: http://blog.tenable.com/
- Tenable-Podcast: http://blog.tenablesecurity.com/podcast/
- Beispielvideos zum Gebrauch: http://www.youtube.com/user/tenablesecurity
- Tenable-Twitterfeed: http://twitter.com/tenablesecurity

Setzen Sie sich mit uns in Verbindung – via E-Mail (support@tenable.com oder sales@tenable.com) oder über unsere Website unter http://www.tenable.com/.

Anhang A: UNIX-Compliancedatei (Beispiel)

Hinweis: Die folgende Datei **`tenable_unix_compliance_template.audit`** ist über das Tenable Support Portal unter https://support.tenable.com/ erhältlich. Die Datei listet die verschiedenen Arten von UNIX-Compliancetests auf, die mit dem UNIX-Compliancemodul von Tenable ausgeführt werden können. Die Datei selbst kann Updates enthalten, die hier nicht wiedergegeben sind.

```
#
# (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>
```

Anhang B: Windows-Compliancedatei (Beispiel)

Hinweis: Die folgende Datei ist über das Tenable Support Portal unter https://support.tenable.com/ erhältlich. Die Datei selbst kann Updates enthalten, die hier nicht wiedergegeben sind. Der Name des Skripts lautet **`financial_microsoft_windows_user_audit_guideline_v2.audit`** und basiert auf gängigen Härtungsleitfäden für die Benutzeradministration. Die Richtlinie prüft, ob eine sinnvolle Kennwortrichtlinie und eine Kontosperrrichtlinie vorhanden sind, und stellt sicher, dass Anmeldeereignisse im Windows-Ereignisprotokoll vermerkt werden.

```
# (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>
```

Anhang C: Konvertierung von XSL-Transformationen nach .audit

Verschiedene Compliancetest-Plugins beruhen auf Audits von XML-Inhalten. Hierzu gehören z. B. Palo Alto-, VMware- und UNIX-Compliancetests. Es empfiehlt sich zur besseren Nutzung dieser Funktionen, sich mit der Erstellung von XSL-Transformationen vertraut zu machen. In einigen Fällen müssen Sie zum Erstellen einer XSL-Transformation ein wenig ausprobieren. Sobald Sie mit der Vorgehensweise vertraut sind, folgt die Konvertierung in eine **.audit** Datei – ein Schritt, der nicht unbedingt intuitiv ist. Mit diesem Anhang erhalten Benutzer eine passende Anleitung für Aufbau und Einsatz von angepassten XSL-Transformationen sowie zu ihrer Konvertierung in **.audit**-Dateien.

Verschiedene Audittests (z. B. AUDIT_XML, AUDIT_VCENTER oder AUDIT_ESX) unterscheiden sich und werden separat eingesetzt, beruhen jedoch auf derselben Logik. Durch einen Einblick in die grundlegende Einsatzweise von XML können Sie die folgenden Grundsätze auch auf andere Plattformen übertragen, die XML verwenden.

Mit dem Utility **xsltproc** können Benutzer in sieben Schritten angepasste **.audit** Dateien für XML-Inhalte erzeugen.

Schritt 1: xsltproc installieren

Überprüfen Sie, ob **xsltproc** auf dem System installiert ist, und installieren Sie es andernfalls. Sie können Vorhandensein und sachgemäße Funktion durch Eingabe des folgenden Befehls überprüfen:

```
[tater@pearl ~]# xsltproc
Einsatz: xsltproc [options] stylesheet file [file ...]
 Optionen:
 --version or -V: show the version of libxml and libxslt used
 --verbose or -v: show logs of what's happening
[..]
```

Schritt 2: Zu verwendende XML-Datei ermitteln

Ermitteln Sie die XML-Datei, die Sie verwenden werden. Überprüfen Sie den Speicherort der Datei und stellen Sie sicher, dass sie XML-Inhalte enthält. Beispiel:

```
[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>
[..]
```

Schritt 3: Mit XSL-Transformationen und XPath vertraut werden

Dieser Vorgang setzt ein grundsätzliches Verständnis von XSL-Transformationen und XPath voraus. Weitere Informationen finden Sie unter:

- w3schools.com: XSLT – Transformation
- w3schools.com: XPath Introduction

Schritt 4: XSL-Transformation erstellen

Beim nächsten Schritt geht es um das Extrahieren der betreffenden Daten aus einer XML-Datei mithilfe von XSL-Transformationen. Erstellen Sie dazu eine XSL-Transformation, mit der Sie die relevanten Daten aus der Datei extrahieren. Im vorliegenden Beispiel extrahieren Sie das Element „name“ aus einer XML-Datei. Die folgende XSLT extrahiert die gewünschten Informationen:

```
<?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>
```

Sobald die XSLT erstellt wurde, speichern Sie sie an einer für den im nächsten Schritt durchzuführenden Test geeigneten Stelle. Dieses Beispiel können Sie als **pa.xsl** speichern.

Wenn Sie mit einer angepassten XSLT in einer .audit-Datei arbeiten, müssen die ersten drei Zeilen sowie letzten zwei Zeilen ignoriert werden. Diese Standardzeilen werden vom Nessus-Plugin **nbin** während der Verarbeitung hinzugefügt. In diesem Beispiel sind die Zeilen 5-8 von Interesse, da sie im Element `AUDIT_XML` oder `AUDIT_REPORTS` benötigt werden.

Der Testablauf in Schritt 5 kann auch beim Erstellen der XSLT zur Bestätigung von Vermutungen und/oder neuen Methoden verwendet werden. Dieser Prozess ist besonders wertvoll, wenn Sie zum ersten Mal mit XSLT oder komplexeren Transformationen arbeiten.

Schritt 5: Funktion der XSLT überprüfen

Überprüfen Sie, ob Ihre XSL-Transformation mit xsltproc funktioniert. Das allgemeine Testformat lautet:

```
/usr/bin/xsltproc {XSLT file} {Source XML}
```

Bei Einsatz der als Beispiel genannten Dateinamen aus den oben stehenden Schritten wird Folgendes ausgegeben. Auf diese Weise erkennen Sie, ob die XSL-Transformation korrekt arbeitet und richtig formatiert ist und die erwarteten Daten ausgegeben werden.

```
[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
```

Schritt 6: XSLT in die .audit-Datei kopieren

Wenn die XSL-Transformation wie vorgesehen funktioniert, kopieren Sie die gewünschten XSLT-Zeilen (in diesem Fall die Zeilen 5-8) in den **`.audit`**-Test.

```
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>"
```

Jede Zeile der angepassten XSL-Transformation muss in doppelten Anführungszeichen in ein eigenständiges **`xsl_stmt`**-Element gesetzt werden. Da das **`xslt_stmt`**-Element doppelte Anführungszeichen zur Verkapselung der `<xsl>`-Statements verwendet, muss allen anderen doppelten Anführungszeichen ein Escape-Zeichen vorangestellt werden. **Das Escape-Zeichen vor den doppelten Anführungszeichen darf keinesfalls vergessen werden, da sonst Fehler bei der Testausführung auftreten**.

```
/usr/bin/xsltproc {XSLT file} {Source XML}
```

Im nächsten Schritt sehen Sie verschiedene Beispiele ordnungsgemäß verwendeter Anführungs- und Escape-Zeichen.

Schritt 7: Abschließendes Audit durchführen

Wenn die ersten sechs Schritte abgeschlossen wurden, verfügen Sie über alle Komponenten zum Aufbau eines Audits:

```
<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>
```

Wissenswertes zu Tenable Network Security

Wenn es um die frühzeitige Erkennung neu entwickelter Sicherheitslücken, Bedrohungen und Compliance-relevanter Risiken geht, verlassen sich mehr als 20.000 Organisationen auf Tenable Network Security. Hierzu gehören neben dem gesamten US-Verteidigungsministerium eine Reihe von Großunternehmen und Regierungsbehörden weltweit. Die Nessus- und SecurityCenter-Lösungen sind nach wie vor branchenführend beim Ermitteln von Sicherheitslücken, beim Verhindern von Angriffen und bei der Erfüllung einer Vielzahl gesetzlicher Vorschriften. Weitere Informationen finden Sie unter www.tenable.com.

GLOBALE UNTERNEHMENSZENTRALE

Tenable Network Security
7021 Columbia Gateway Drive
Suite 500
Columbia, MD 21046, USA
+1 410.872.0555
www.tenable.com