RSA NETWITNESS® SUITE

Befehlszeilenoberflächen-Benutzerhandbuch

für Version 11.0

Kontaktinformationen

Der RSA-Link unter https://community.rsa.com enthält eine Wissensdatenbank, in der allgemeine Fragen beantwortet und Lösungen für bekannte Probleme, Produktdokumentationen, Communitydiskussionen und Vorgangsmanagement bereitgestellt werden.

Marken

Eine Liste der RSA-Marken finden Sie unter germany.emc.com/legal/emc-corporation-trademarks.htm#rsa.

Lizenzvereinbarung

Diese Software und die zugehörige Dokumentation sind Eigentum von EMC und vertraulich. Sie werden unter Lizenz bereitgestellt und dürfen nur gemäß den Bedingungen der betreffenden Lizenz und unter Einschluss des untenstehenden Copyright-Hinweises verwendet und kopiert werden. Diese Software und die Dokumentation sowie alle Kopien dürfen anderen Personen nicht überlassen oder auf andere Weise zur Verfügung gestellt werden.

Dabei werden keine Ansprüche oder Eigentumsrechte an der Software oder Dokumentation oder Rechte an geistigem Eigentum daran übertragen. Die unberechtigte Nutzung oder die Vervielfältigung dieser Software und der Dokumentation kann zivil- und/oder strafrechtlich verfolgt werden.

Diese Software kann ohne Vorankündigung geändert werden und sollte nicht als Verpflichtung seitens EMC ausgelegt werden.

Drittanbieterlizenzen

Dieses Produkt kann Software enthalten, die von anderen Anbietern als RSA entwickelt wurde. Der Text der Lizenzvereinbarungen, die sich auf Drittanbietersoftware in diesem Produkt beziehen, ist auf der Produktdokumentationsseite auf RSA Link verfügbar. Mit der Verwendung dieses Produkts verpflichtet sich der Benutzer zur uneingeschränkten Einhaltung der Bedingungen der Lizenzvereinbarungen.

Hinweis zu Verschlüsselungstechnologien

Dieses Produkt kann Verschlüsselungstechnologie enthalten. In vielen Ländern ist die Verwendung, das Importieren oder Exportieren von Verschlüsselungstechnologien untersagt. Die aktuellen Bestimmungen zum Verwenden, Importieren und Exportieren sollten beim Verwenden, Importieren und Exportieren dieses Produkts eingehalten werden.

Verteilung

EMC ist der Ansicht, dass die Informationen in dieser Veröffentlichung zum Zeitpunkt der Veröffentlichung korrekt sind. Diese Informationen können jederzeit ohne vorherige Ankündigung geändert werden.

Februar 2018

Inhalt

Zugreifen auf NwConsole und Hilfe

RSA NetWitness Console, auch als NwConsole-Konsole bezeichnet, ist eine Multiplattform-Terminal-Anwendung, die leistungsstarke Tools und Befehlszeilenzugriff auf Core-Services bietet, z. B. auf Decoder, Log Decoder, Concentrator, Broker und Archiver. Während die meisten Benutzer ihre Aufgaben und Ermittlungen über die NetWitness Suite-Benutzeroberfläche durchführen, benötigen einige erfahrene Benutzer, z. B. Administratoren und Entwickler, direkten Zugriff auf die Services, ohne über die Benutzeroberfläche zuzugreifen. NwConsole ermöglicht das Eingeben von Befehlen über die Befehlszeile sowie das Ausführen mehrerer Befehle über eine Datei.

In diesem Thema wird beschrieben, wie Sie auf NwConsole zugreifen und die interne Hilfe in NwConsole anzeigen.

Umfassende Hilfeinformationen sind in der RSA Security Analytics-Konsole verfügbar, auch als NwConsole bezeichnet. Sie können über die Security Analytics-Befehlszeile auf diese Hilfe zugreifen.

Voraussetzungen

Die Anwendung NwConsole ist auf allen NetWitness Suite-Appliances installiert. Für die Verbindung zu und Interaktion mit einem Core-Service können Sie sie auch unter Windows, Mac und CentOS installieren.

NwConsole ist über die Befehlszeile auf NetWitness Suite-Appliances verfügbar. Wenn Sie remote auf eine Core-Appliance zugreifen, muss die RSA NetWitness Console-Anwendung auf einem Windows-, Mac- oder CentOS-Rechner installiert sein. Wenden Sie sich an die RSA-Kundenbetreuung, um das Installationsprogramm für die RSA NetWitness Console zu erhalten.

Zugreifen auf NwConsole

Geben Sie zum Ausführen von NwConsole über die Befehlszeile auf einer NetWitness Suite-Appliance oder auf einem Terminal-Emulator in der <$>-Eingabeaufforderung `NwConsole` (Linux) oder `nwconsole` (Windows) ein. Der tatsächliche Befehl lautet `NwConsole`, aber unter Windows wird nicht zwischen Groß- und Kleinschreibung unterschieden. Die RSA NetWitness Console wird wie im folgenden Beispiel dargestellt angezeigt.

```
Last login: Thu Sep 24 14:00:42 on console
usxx<username>m1:~ <username>$ NwConsole
RSA NetWitness Suite Console 10.6.0.0.6105
Copyright 2001-2015, RSA Security Inc.  All Rights Reserved.
```

```
Type "help" for a list of commands or "man" for a list of manual pages.
>
```

Hilfe anzeigen

NwConsole enthält eine Hilfe zu den einzelnen Befehlen sowie zu bestimmten Themen.

Achtung: Die neuesten Informationen finden Sie in den Informationen zu Befehlen und die Hilfethemen in NwConsole.

Anzeigen einer Liste der Befehle

Geben Sie zum Anzeigen einer Liste der verfügbaren Befehle und ihrer Beschreibungen in der (>)-Eingabeaufforderung `help` ein. Im folgenden Beispiel ist eine Liste verfügbarer Befehle gezeigt.

```
> help
Local commands:
    avro2nwd      - Convert AVRO files to NWD files
    avrodump      - Display schema and contents of AVRO file (for debugging)
    blockspeed    - Tests various write block sizes to determine best setting
    compileflex   - Compile all flex parsers in a directory
    createflex    - Create a flex parser that matches tokens read from a file
    dbcheck       - Perform a database integrity check over one or more
                    session, meta, packet, log or stat db files
    diskspeed     - Measures the speed of the disk(s) mounted at a specified
                    directory
    echo          - Echos the passed in text to the terminal
    encryptparser - Encrypt all parsers in a directory
    feed          - Create and work with feed files
    fmanip        - Manipulate a file with XOR and check for embedded PEs
    hash          - Creates or verifies hashes of database files
    help          - Provides help information for recognized console commands
    history       - Displays, erases or executes a command in the command
                    history
    httpAggStats  - Tests HTTP aggregation and reports statistics as it
                    continues
    log           - Perform operations on a log database
    logParse      - Parse line delimited logs on stdin and post results to
                    stdout
    logfake       - Create a fake log pcap file
```

```
    lua           - Execute a lua script
    makec3        - Generate C3 Test Data
    makepcap      - Convert packet database files to pcap or log files
    man           - Displays a list of topics or opens a specific manual page
                    on a topic
    metaspeed     - Tests read performance over an existing meta db
    netbytes      - Display statistics on network interface utilization
    nwdstrip      - Convert full NWD file into just session and meta file
    pause         - Wait for user input when running a script file
    reindex       - reindex a collection
    sdk           - Execute SDK commands based on the C SDK library, type "sdk
                    help" for more information
    sleep         - Sleeps for the specified milliseconds
    timeout       - Globally change the timeout for waiting for a response from
                    a service
    tlogin        - Open a trusted SSL connection to an existing service
    topQuery      - Returns the top N longest running queries from the audit
                    log (either a file or from the log API)
    vslice        - Validate index slices

Remote commands (executed on the connected service, see "login"):
    login         - Connect to a remote service.  Once connected, type help to
                    see commands available for remote execution.

For detailed help, type "help <command>"
>
```

Ausführliche Hilfe zu einem Befehl anzeigen

Geben Sie zum Anzeigen detaillierter Informationen zu einem Befehl `help <command>` ein. Im folgenden Beispiel ist die Hilfe für den Befehl `logParse` nach Eingabe von `help logParse` gezeigt.

```
For detailed help, type "help <command>"
> help logParse
Usage: logParse {in=<pathname>} {indir=<pathname>} [out=<pathname>]
               [content=<c2|c3>] [device=<device,[device...]>]
               [path=<log-parsers-config-path>] [metaonly] [srcaddr=<src
               address>] [srcaddrfile=<filename,IP Address>]
Parse line delimited logs on stdin and post results to stdout
```

```
    in          - The input source file. "in=stdin" means interactive typing of
                  log.
    indir       - The input source files parent directory
    out         - The output file or output file parent directory if input is
                  set by indir. If not specified, use stdout as output.
    content     - Content version, either c2 or c3. Default is c2.
    device      - Comma delimited device list specifying devices that is
                  enabled. Default enable all devices.
    path        - The logparsers configuration path. Default will find
                  configuration file like logdecoder.
    metaonly    - The output will only contains parsed meta, otherwise will
                  print log message after metas.
    srcaddr     - The source address of the all the logs
    srcaddrfile - The source address for logs in one input file, in the format
                  filename,ipaddress
>
```

Anzeigen einer Liste von Hilfethemen

Geben Sie zum Anzeigen einer Liste mit Hilfethemen `man` ein. Im folgenden Beispiel ist eine Liste mit Hilfethemen gezeigt.

```
> man
List of topics:

    Introduction
    Connecting to a Service
    Monitoring Stats
    Feeds
    Converting Packet DB Files to PCAP
    Packets
    Verifying Database Hashes
    SDK Content
    SDK Content Examples
    Troubleshooting

Type "man <topic>" for help on a specific topic, partial matches are acceptable
>
```

Anzeigen eines bestimmten Hilfethemas

Geben Sie zum Anzeigen von Hilfe zu einem bestimmten Thema `man <topic>` ein. Im folgenden Beispiel ist das Hilfethema „Pakete“ nach Eingabe von `man Packets` gezeigt.

```
Type "man <topic>" for help on a specific topic, partial matches are acceptable
> man Packets
                                    Packets
                                    =======


The *packets* command can be used to generate a pcap or log file based on a
list of Session IDs, a time period or a where clause.  The command is quite
flexible and can be used on any running service that has access to the raw
data from a downstream component.  Before running the command, you must first
*login* to a service and then change directory to the appropriate sdk node,
(e.g., "cd /sdk").  Unlike the *makepcap* command, which only works on the
local file system, this command is meant to be used on a remote service.

    login ...
    cd /sdk
    packets where="service=80 && time='2015-03-01 15:00:00'-'2015-03-01
    15:10:00'" pathname="/tmp/march-1.pcap"
Write 10 minutes of HTTP only packets from March 1st, to the file
/tmp/march-1.pcap.  All times are in UTC.

    packets time1="2015-04-01 12:30:00" time2="2015-04-01 12:35:00"
    pathname=/media/sdd1/packets.pcap.gz
Write all packets between the two times to a gzip compressed file at
/media/sdd1/packets.pcap.gz

    packets time1="2015-04-01 12:30:00" time2="2015-04-01 12:35:00"
    pathname=/media/sdd1/mylogs.log
Write all logs between the two times to a plaintext file at
/media/sdd1/mylogs.log.  Any pathname ending with .log indicates that the
format of the output file should be plaintext line-delimited logs.
>
```

Achtung: Die neuesten Informationen finden Sie in den Informationen zu Befehlen und die Hilfethemen in NwConsole.

Beenden von NwConsole

Geben Sie zum Beenden der Anwendung NwConsole in der Befehlszeile `quit` ein.

Grundlegende Befehlszeilenparameter und Bearbeiten

NwConsole ist wie ein Schweizer Taschenmesser. Die Befehlszeilenoberfläche umfasst alle möglichen Tools, die nicht auf den ersten Blick sichtbar sind. NwConsole ist eine Multiplattformanwendung. Die ausführbaren Dateien sind für CentOS (wird bereits auf Appliances ausgeliefert), Windows und Mac verfügbar.

Grundlegende Befehlszeilenparameter

Im Folgenden finden Sie einige grundlegende Befehlszeilenparameter:

- So führen Sie eine Reihe von Befehlen aus einer Datei aus:
  ```
  NwConsole -f /tmp/somefile.script
  ```
- So übergeben Sie eine Liste von Befehlen über die Befehlszeile:
  ```
  NwConsole -c <command1> -c <command2> -c <command3>
  ```
  Dies ist nicht zwangsläufig zu empfehlen, außer für sehr einfache Skripte. Der Bash-Interpreter kann Zeichenfolgen in Anführungszeichen in sinnlose Zeichenfolgen zerlegen, wenn Sie sie nicht ordnungsgemäß mit Escapezeichen kennzeichnen. Wenn bei der Übergabe über die Befehlszeile unklare Fehler auftreten, wechseln Sie zum Lesen aus einer Datei, um zu überprüfen, ob die Probleme dadurch behoben werden.
- In der Regel wird die Konsole nach der Ausführung von Befehlen über eine Datei oder die Befehlszeile beendet. Wenn die interaktive Aufforderung geöffnet bleiben soll, nachdem die Befehle ausgeführt wurden, fügen Sie in der Befehlszeile `-i` hinzu.
- Selbstverständlich können Sie NwConsole auch einfach ausführen und die Befehle im Konsolenfenster eingeben.

Bearbeiten von Zeilen

Sie können die Tasten in der folgenden Tabelle verwenden, wenn Sie einen Befehl bearbeiten.

Schlüssel	Beschreibung
Strg-U	Löscht die aktuelle Zeile
Strg-W	Löscht das Wort, auf dem sich der Cursor befindet.
Strg-A	Bewegt den Cursor an den Anfang der Zeile

Schlüssel	Beschreibung
Strg-E	Bewegt den Cursor an das Ende der Zeile
Pfeil nach oben	Zeigt den zuvor ausgeführten Befehl an
Pfeil nach unten	Zeigt den Befehl an, der nach dem aktuellen Befehl ausgeführt wird (gilt nur, wenn Pfeil nach oben gedrückt wurde)
Pfeil nach links	Bewegt den Cursor auf das vorherige Zeichen
Pfeil nach rechts	Bewegt den Cursor auf das nächste Zeichen
Registerkarte	Bietet kontextabhängige Vervollständigung der meisten Befehle und ihrer Parameter. Die Tabulatortaste ist sehr hilfreich für die Bearbeitung. Beispiel: Wenn Sie das Hilfethema *Herstellen einer Verbindung mit einem Service* anzeigen möchten, können Sie in der Befehlszeile `mancon` eingeben und dann die Taste `Tab` drücken. NwConsole vervollständigt den Befehl für Sie: `man Connecting to a Service` Drücken Sie die Eingabetaste, um den Befehl auszuführen und das Thema anzuzeigen.
`history`	Zeigt eine nummerierte Liste der vorherigen Befehle an
`history execute=#`	Ein vorheriger Befehl wird ausgeführt. Dies entspricht der Eingabe `!#`. Beispielsweise führt `!1` den vorherigen Befehl aus.
`history clear`	Löscht den gesamten Befehlsverlauf
`history erase=#`	Löscht einen bestimmten Befehl aus dem Verlaufspuffer. Der Verlauf wird automatisch von einer Sitzung zur nächsten gespeichert.

Herstellen der Verbindung zu einem Service

Wenn Sie eine Verbindung zu einem Security Analytics Core-Service (Decoder, Concentrator, Broker, Archiver usw.) herstellen und dann mit diesem interagieren möchten, müssen Sie zunächst den Befehl `login` ausführen. Sie benötigen ein Konto für den Service. Sie können jederzeit „`type help` login“ eingeben, um weitere Informationen zu erhalten. Die Syntax des Befehls `login` lautet wie folgt:

```
login <hostname>:<port>[:ssl] <username> [password]
```

Beispiel: **login 10.10.1.15:56005:ssl someuser**

Wenn Sie das Passwort nicht eingeben, wird es angefordert und das entsprechende Passwort-Masking wird durchgeführt.

Wenn Sie die richtige Vertrauenseinstellung zwischen NwConsole und dem Endpunkt eingerichtet haben, können Sie den Befehl `tlogin` verwenden und müssen kein Passwort eingeben. Das Einrichten der Vertrauenseinstellungen geht über den Umfang dieser Dokumentation hinaus, aber es umfasst das Hinzufügen des SSL-Zertifikats von NwConsole auf dem Endpunkt über den Befehl `send /sys peerCert op=add --file-data=<pathname of cert>`. Bevor Sie ein Peer-Zertifikat für nachfolgende vertrauenswürdige Anmeldungen hinzufügen können, müssen Sie zunächst eine normale Anmeldung mit den entsprechenden Berechtigungen durchführen.

Sobald die Verbindung hergestellt ist, können Sie über ein virtuelles Dateisystem mit dem Endpunktservice interagieren. Anstelle von Dateien werden die Nodes des Services angezeigt. Bei einigen Nodes handelt es sich um Ordner mit untergeordneten Nodes, die eine hierarchische Struktur bilden. Jeder Node dient einem Zweck und alle unterstützen eine Untergruppe von Befehlen wie `info` und `help`. Die Meldung `help` gibt Informationen über die Befehle zurück, die jeder Node unterstützt. Wenn Sie sich das erste Mal anmelden, befinden Sie sich auf dem Stamm-Node. Dabei handelt es sich um den Pfad `/`, genau wie auf einem Linux- oder Mac-System. Geben Sie zum Anzeigen einer Liste von Nodes unter `/` den Befehl `ls` ein.

Alle Services verfügen über Nodes wie `sys` und `logs`. Für die Interaktion mit der **/logs**-API können Sie zunächst den Befehl `help` an den **/logs**-Node senden. Dafür müssen Sie die Meldung `send` mit der folgenden Syntax verwenden:

```
Usage: send {node pathname} {message name} [name=value [name=value]]

            [--file-data=<pathname>] [--string-data=<text>] [--binary-data=<text>]
            [--output-pathname=<pathname>] [--output-append-pathname=<pathname>]
            [--output-format={text,json,xml,html}]
Sends a command to a remote pathname.  For remote help, use "send <pathname>help"
```

```
for details.

    pathname                 - The node pathname to retrieve information on
    message                  - The command (message) to send
    parameters               - Zero or more name=value parameters for the command
    --file-data              - Loads data from a file and send as either a BINARY
                               message or as a PARAMS_BINARY message if other
                               parameters exist
    --string-data            - Sends text as a STRING message type
    --binary-data            - Send text as either a BINARY message type or as a
                               PARAMS_BINARY message type if other parameters
                               exist
    --output-pathname        - Writes the response output to the given pathname,
                               overwriting any existing file
    --output-append-pathname - Writes the response output to the given pathname,
                               will append output to an existing file
    --output-format          - Writes the response in one of the given formats,
                               the default is text
```

Wenn Sie eine `help`-Meldung senden möchten, senden Sie also Folgendes:

```
send /logs help
```

Und Ihre Antwort würde wie folgt aussehen:

```
    description: A container node for other node types
    security.roles: everyone,logs.manage
    message.list: The list of supported messages for this node
    ls:  [depth:<uint32>] [options:<string>] [exclude:<string>]
    mon:  [depth:<uint32>] [options:<uint32>]
    pull:  [id1:<uint64>] [id2:<uint64>] [count:<uint32>] [timeFormat:<string>]
    info:
    help:  [msg:<string>] [op:<string>] [format:<string>]
    count:
    stopMon:
    download:  [id1:<uint64>] [id2:<uint64>] [time1:<date-time>] [time2:<date-
time>] op:<string>
  [logTypes:<string>] [match:<string>] [regex:<string>] [timeFormat:<string>]
[batchSize:<uint32>]
    timeRoll:  [timeCalc:<string>] [minutes:<uint32>] [hours:<uint32>]
[days:<uint32>] [date:<string>]
```

Um weitere Informationen über eine bestimmte Nachricht oder einen Befehl zu erhalten, können Sie den `msg=<message name>` bei dem Hilfebefehl als Parameter angeben. So rufen Sie z. B. die Hilfe zur Meldung `pull` auf:

```
send /logs help msg=pull
```

```
    pull: Downloads N log entries
    security.roles: logs.manage
    parameters:
    id1 - <uint64, optional> The first log id number to retrieve, this is mutually
exclusive with id2
    id2 - <uint64, optional> The last log id number that will be sent, defaults to
most recent log
 message when id1 or id2 is not sent
    count - <uint32, optional, {range:1 to 10000}> The number of logs to pull
    timeFormat - <string, optional, {enum-one:posix|simple}> The time format used
in each log message,
 default is posix time (seconds since 1970)
```

Die integrierte Hilfemeldung gibt zurück, dass dieser Befehl die letzten n Protokolleinträge abruft, wenn Sie id1 und id2 deaktiviert lassen. So zeigen Sie die letzten 10 Protokolleinträge dieses Service an:

```
send /logs pull count=10 timeFormat=simple
```

Fast alle Befehle auf dem Service weisen dieses einfache Format auf. Die einzigen Befehle, bei denen dies nicht der Fall ist, sind diejenigen, die kompliziertere Handshakes erfordern, wie das Importieren eines PCAP in einen Decoder. Verwenden Sie zum Importieren eines PCAP den NwConsole-Befehl `import`, der das komplizierte Kommunikationskanal-Handshaking übernimmt.

Einige Parameter sind spezifisch für den NwConsole-Befehl `send` und werden nicht an den Service gesendet. Sie können diese Parameter verwenden, um das Ausgabeformat der Antwort zu ändern, die Antwort in eine Datei zu schreiben oder eine Datei vom lokalen Computer zu lesen und an den Service zu senden. Die lokalen Parameter für den NwConsole-Befehl `send` beginnen alle mit zwei Bindestrichen (--).

- `--output-format`: Dieser Parameter ändert die normale Ausgabe des Befehls von Klartext in einen der folgenden Typen: JSON, XML oder HTML.
- `--output-pathname`: Statt die Ausgabe auf dem Terminal zu schreiben, wird diese in den angegebenen Pfadnamen geschrieben (kürzt vorhandene Dateien).

- `--output-append-pathname`: Dies entspricht `--output-pathname` mit dem Unterschied, dass die Ausgabe an eine vorhandene Datei angehängt wird (oder die Datei erstellt wird, wenn sie nicht vorhanden ist).
- `--file-data`: Liest in einer Datei und verwendet diese als Befehlsnutzlast. Dies ist nützlich für Befehle wie `/sys fileEdit`. Das folgende Beispiel zeigt, wie Sie eine aktualisierte **index-concentrator-custom.xml**-Datei mithilfe von NwConsole senden können:

```
send /sys fileEdit op=put filename=index-concentrator-custom.xml --file-
data="/Users/user/Documents/index-concentrator-custom.xml"
```

- `--file-format`: Dieser Parameter zwingt NwConsole beim Lesen einer Eingabedatei mit `--file-data`, die Datei als einen bestimmten Eingabetyp zu interpretieren. Die folgenden Enumeration sind zulässig: `binary`, `params`, `param-list`, `string` und `params-binary`. Beispiel: Um eine Datei mit Anwendungsregeln (`*.nwr`) an einen Decoder zu senden, können Sie folgenden Befehl verwenden:

```
send /decoder/config/rules/application replace --file-
data=/path/rules.nwr --file-format=param-list
```

- `--string-data`: Sendet die Befehlsnutzdaten als Zeichenfolge anstelle einer Liste von Parametern.
- `--binary-data`: Sendet die Befehlsnutzdaten als Binärdaten anstelle einer Liste von Parametern.

Beispiel-Streaming-Abfrage zu einer JSON-Datei (möglicherweise großer Ergebnissatz):

```
send /sdk query size=0 query="select * where service=80 && time='2015-
03-05 13:00:00'-'2015-03-05 13:59:59'" --output-format=json --output-
pathname=/tmp/query.json
```

Beim Befehl „send“ sollte beachtet werden, dass es standardmäßig zu einem Timeout von 30 Sekunden beim Warten auf eine Antwort kommt. Bei einigen Befehlen (wie der obigen Abfrage) dauert es möglicherweise länger, bis Ergebnisse empfangen werden. Um einen vorzeitigen clientseitigen Timeout zu vermeiden, können Sie den Befehl`timeout [secs]` verwenden, um die Wartezeit erhöhen. Beispiel: `timeout 600` würde 10 Minuten auf eine Antwort warten, bevor es zu einem Timeout kommt. Nach der Ausführung wird das für alle nachfolgenden Befehle wirksam.

Um in der virtuellen Node-Hierarchie des Services zu navigieren, können Sie den Befehl `cd` wie in einer Befehls-Shell verwenden. Dies deckt die Grundlagen des Verbindens und der Interaktion mit einem Service ab. Sobald die Verbindung besteht, können Sie mit dem Befehl `help` alle Befehle auflisten lassen, die Sie zur Interaktion mit dem Endpunkt verwenden können. Diese Befehle werden nicht angezeigt, wenn Sie nicht mit einem Endpunkt verbunden sind.

Monitoring von Statistiken

Sie können NwConsole verwenden, um Änderungen an den Statistiken eines Services in Echtzeit nachzuverfolgen. Beachten Sie jedoch, dass dies zu einer GROSSEN Ausgabemenge führen kann. Wenn Sie zu viele Nodes überwachen, ist die Bildlaufgeschwindigkeit zu groß, um nützlich sein.

Einfaches Beispiel: Wenn Sie sich bei einem Decoder anmelden, können Sie die Erfassungsrate in Echtzeit überwachen. Dazu geben Sie die folgenden Befehle ein, nachdem Sie eine Verbindung zu einem Decoder hergestellt haben:

```
decoder/stats
```

```
mon capture.rate
```

Das ist alles, was Sie tun müssen. Jetzt wird bei jeder Änderung der Erfassungsrate eine Ausgabe im Konsolenfenster angezeigt.

Sie können ein weiteres Monitoring hinzufügen:

```
mon capture.avg.size
```

Jetzt werden diese beiden Statistiken überwacht und die Werte werden bei einer Änderung ausgegeben. Möglicherweise haben Sie bemerkt, dass die Ausgabe des ersten Monitorings beim Eingeben des zweiten Befehls die Anzeige unleserlich gemacht hat. Dies ist das Problem beim Monitoring von Statistiken. Das Monitoring ist darauf ausgelegt, dass nach Eingabe des ersten Befehls nur noch die Statistiken beobachtet werden.

Sie können das Monitoring jedoch beenden, indem Sie `delmons` eingeben und die **Eingabetaste** drücken. Ignorieren Sie einfach die Ausgabe, während Sie den Befehl eingeben, und Sie kehren zu einer normalen Eingabeaufforderung zurück. Wenn Sie viele Statistiken auf einmal überwachen möchten, können Sie nur den Pfad des Ordners der übergeordneten Statistik angeben und es werden alle Statistiken darunter überwacht. Wenn Sie z. B. `mon /decoder/stats` oder `mon .` (diese sind gleichwertig) eingeben, wird alles überwacht. Stellen Sie sich dabei auf eine große Menge Ausgaben ein. Denken Sie daran, `delmons` einzugeben, wenn der Bildlauf zu schnell ist.

Nützliche Befehle

Die folgenden NwConsole-Befehle sind hilfreich bei der Interaktion mitNetWitness-ServerCore-Services:

- **feed**: Ermöglicht das Erstellen von und Arbeiten mit Feeddateien.
- **makepcap**: Konvertiert Paketdatenbankdateien in PCAP
- **packets**: Ruft Pakete oder Protokolle vom angemeldeten Service ab
- **hash**: Erstellt oder überprüft Hashes von Datenbankdateien

In den folgenden Abschnitten sowie auf den Seiten mit Informationen zur Hilfe und zu Themen zu NwConsole finden Sie zusätzliche Informationen.

Feeds

Der Befehl `feed` bietet mehrere Dienstprogramme für die Erstellung und Untersuchung von Feeddateien. Eine Feeddatei enthält die Definition und die Daten eines einzigen Feeds in einem Format, das von einem Decoder oder Log Decoder für effizientes Laden vorkompiliert wurde. Eine vollständige Referenz zu Feeddefinitionen finden Sie unter **Feeddefinitionsdatei** im *Decoder- und Log Decoder-Konfigurationsleitfaden*.

create

```
feed create <definitionfile> [-x <password>]
```

Der Befehl `feed create` erzeugt Feeddateien für jeden in einer Feeddefinitionsdatei definierten Feed. Eine Definitionsdatei ist ein XML-Dokument, das eine oder mehrere Definitionen enthält. Jede Feeddefinition gibt eine Datendatei und die Struktur der Datendatei an. Die daraus resultierenden Feeddateien werden im selben Verzeichnis wie die Definitionsdatei mit demselben Namen wie die Datendatei erstellt. Die Erweiterung wird aber in **.feed** geändert (z. B. wird **datafile.csv** zu **datafile.feed**). Vorhandene Dateien mit dem Zielnamen werden ohne Bestätigungsanforderung überschrieben.

```
$ ls
example-definition.xml    example-data.csv
$ NwConsole
RSA NetWitness Console 10.5.0.0.0
Copyright 2001-2015, RSA Security Inc.  All Rights Reserved.

Type "help" for a list of commands or "man" for a list of manual pages.
> feed create example-definition.xml
Creating feed Example Feed...
```

```
done.  2 entries, 0 invalid records
All feeds complete.
> quit
$ ls
example-definition.xml    example-data.feed    example-data.csv
$
```

Optional können Feeddateien mithilfe der Option `-x` gefolgt von einem Passwort von mindestens 16 Zeichen Länge (ohne Leerzeichen) verschleiert werden. Dies wird auf alle Feeds in der Definitionsdatei angewandt. Zusätzlich zu der Feeddatei wird eine Tokendatei für jede Feeddatei erzeugt. Die Tokendatei muss mit der entsprechenden Feeddatei bereitgestellt werden.

```
feed create example-definition.xml -x 0123456789abcdef
```

stats

```
feed stats <feedfile>
```

Der Befehl `feed stats` bietet zusammenfassende Informationen über eine vorhandene, nicht verschleierte Feeddatei. Bei Angabe einer verschleierten Feeddatei tritt ein Fehler auf.

```
> feed stats example.feed
 Example Feed stats:
 version  : 0
 keys count  : 1
 values count: 2
 record count: 2
 meta key  : ip.src/ip.dst
 language keys:
 alert    Text
```

dump

```
feed dump <feedfile> <outfile>
```

Der Befehl `feed dump` erzeugt eine normalisierte Liste mit Schlüsselwertpaaren zu einer nicht verschleierten Feeddatei. Sie können die resultierende Datei zur Validierung einer Feeddatei oder als Hilfe bei der Ermittlung verwenden, welche Datensätze bei Erstellung des Feeds als ungültig betrachtet wurden. Bei Angabe einer verschleierten Feeddatei tritt ein Fehler auf. Wenn `outfile` vorhanden ist, wird der Befehl abgebrochen, ohne die vorhandene Datei zu überschreiben.

```
feed dump example.feed example-dump.txt
```

Konvertieren von Paket-DB-Dateien in PCAP

Sie können den Befehl `makepcap` verwenden, um schnell alle Paket-DB-Datei in eine allgemeine PCAP-Datei zu konvertieren. Dabei wird die Reihenfolge der Erfassungszeit beibehalten. Dieser Befehl bietet viele Optionen (siehe`help makepcap`), ist aber leicht anzuwenden. Alles was Sie für den Einstieg brauchen ist das Paket-DB-Verzeichnis (über den Parameter `source=<pathname>`).

Hinweis: Sie müssen den Decoder- oder Archiver-Service vor der Ausführung dieses Befehls beenden. Wenn Sie eine PCAP-Datei erzeugen möchten, während der Dienst ausgeführt wird, lesen Sie die Informationen zum Befehl `packets`.

```
makepcap source=/var/netwitness/decoder/packetdb
```

Dieser Befehl konvertiert jede Paket-DB-Datei in eine entsprechende PCAP-Datei im selben Verzeichnis. Wenn die Festplatte fast voll ist, lesen Sie die Informationen zum nächsten Befehl.

```
makepcap source=/var/netwitness/decoder/packetdb
dest=/media/usb/sde1
```

Dieser Befehl schreibt alle Ausgabe-PCAPs in das Verzeichnis unter **/media/usb/sde1**.

```
makepcap source=/var/netwitness/decoder/packetdb
dest=/media/usb/sde1 filenum=4-6
```

Dieser Befehl konvertiert nur die Dateien mit der Nummer 4 bis 6 und überspringt alle anderen Dateien. Mit anderen Worten, es werden die folgenden Paket-DB-Dateien konvertiert: **packet-000000004.nwpdb**, **packet-000000005.nwpdb** und **packet-000000006.nwpdb**.

```
makepcap source=/var/netwitness/decoder/packetdb time1="2015-03-
01 14:00:00" time2="2015-03-02 07:30:00" fileType=pcapng
```

Mit diesem Befehl werden nur Pakete mit einem Zeitstempel zwischen dem 1. März 2015, 14:00 Uhr und dem 2. März 2015 vor oder um 7:30 Uhr extrahiert. Die Datei wird als pcapng in dasselbe Verzeichnis wie die Quelle geschrieben. Alle Zeitstempel werden in UTC angegeben.

Pakete

Sie können den Befehl `packets` verwenden, um basierend auf einer Liste von Sitzungs-IDs, einem Zeitraum oder einer Where-Klausel eine PCAP- oder Protokolldatei zu erzeugen. Dieser Befehl ist sehr flexibel, sodass Sie ihn auf jeden laufenden Service anwenden können, der Zugriff auf Rohdaten einer Downstream-Komponente hat. Bevor Sie den Befehl ausführen, müssen Sie zunächst den Befehl `login` für einen Service ausführen und dann das Verzeichnis in den entsprechenden SDK-Node ändern (z. B. `cd /sdk`). Im Gegensatz zum Befehl `makepcap`, der nur auf dem lokalen Dateisystem arbeitet, verwenden Sie diesen Befehl für einen Remoteservice.

```
login ...
```

```
cd /sdk

packets where="service=80 && time='2015-03-01 15:00:00'-'2015-03-01 15:10:00'"
pathname="/tmp/march-1.pcap"
```

Dieser Befehl schreibt 10 Minuten reine HTTP-Pakete vom 1. März in die Datei **/tmp/march-1.pcap**. Alle Zeiten werden in UTC angegeben.

```
packets time1="2015-04-01 12:30:00" time2="2015-04-01 12:35:00"
pathname=/media/sdd1/packets.pcap.gz
```

Dieser Befehl schreibt alle Pakete zwischen den beiden Zeiten in eine komprimierte GZIP-Datei unter **/media/sdd1/packets.pcap.gz**.

```
packets time1="2015-04-01 12:30:00" time2="2015-04-01 12:35:00"
pathname=/media/sdd1/mylogs.log
```

Dieser Befehl schreibt alle Protokolle zwischen den beiden Zeiten in eine Klartextdatei unter **/media/sdd1/mylogs.log**. Mit **.log** endende Pfadnamen zeigen an, dass die Ausgabedatei als durch Zeilen getrennte Klartextprotokolle formatiert sein sollte.

Überprüfen von Datenbank-Hashes

Standardmäßig schreibt Archiver für jede DB-Datei, die geschrieben wird, eine XML-Datei. Diese XML-Datei endet mit der Erweiterung **.hash** und enthält einen Hash der Datei zusammen mit anderen relevanten Informationen. Sie können den Befehl `hash` verwenden, um sich zu vergewissern, dass die DB-Datei nicht manipuliert wurde, indem Sie den in der XML-Datei gespeicherten Hash lesen und dann ein erneutes Hashing für die DB-Datei durchführen, um zu überprüfen, ob der Hash gültig ist.

```
hash op=verify
hashfile=/var/netwitness/archiver/database0/alldata/packetdb/packet-000004880.nwpdb.hash
```

Dieser Befehl überprüft, ob die Paket-DB-Datei **packet-000004880.nwpdb** noch dem Hash in der XML-Datei **packet-000004880.nwpdb.hash** entspricht. Aussicherheitsgründen sollte die Hash-Datei an einem anderen Ort gespeichert werden, um zu verhindern, dass die XML-Datei manipuliert werden kann (z. B. auf nur einmal beschreibbaren Medien). Für den Hash-Befehl selbst ist es unerheblich, wo sie gespeichert ist.

Befehl „sdk content“

Einer der leistungsstärksten Befehle in NwConsole ist `sdk content`. Er umfasst zahlreiche Optionen, mit denen in Bezug auf das Extrahieren von Inhalten aus dem NetWitness Suite-Core-Stack fast keine Wünsche offen bleiben. Sie können ihn verwenden, um PCAP- und Protokolldateien zu erstellen oder um Dateien aus Netzwerksitzungen zu extrahieren (z. B. alle Bilder aus E-Mail-Sitzungen zu sammeln). Es ist möglich, mit dem Befehl Dateien anzuhängen, vor der Erstellung einer Datei eine Maximalgröße festzulegen und Dateien automatisch zu bereinigen, wenn das Verzeichnis zu groß wird. Es können im Hintergrund Abfragen für die Suche nach neuen Sitzungen durchgeführt werden. Abfragen werden in verwaltbare Gruppen unterteilt und diese Vorgänge werden automatisch ausgeführt. Wenn die Gruppe erschöpft ist, wird eine erneute Abfrage durchgeführt, um einen neuen Datensatz für weitere Vorgänge zu erhalten. Die Liste der Optionen für den Befehl „sdk content“ ist sehr umfangreich.

Da der Befehl so viele Optionen umfasst, enthält dieses Dokument Beispielbefehle für verschiedene Anwendungsfälle.

Bevor Sie den Befehl `sdk content` ausführen können, müssen Sie zunächst einige andere Befehle ausführen (z. B. Anmeldung bei einem Service). Hier einige Beispiele:

- Verbinden Sie sich zunächst mit einem Service:
  `sdk open nw://admin:netwitness@10.10.25.50:50005`
- Wenn Sie eine Verbindung über SSL herstellen müssen, verwenden Sie das Protokoll nws:
  `sdk open nws://admin:netwitness@10.10.25.50:56005`
- Denken Sie daran, dass Sie eine URL übergeben und diese korrekt URL-kodiert werden muss. Wenn das Passwort `p@ssword` lautet, sieht die URL wie folgt aus: `sdk open nw://admin:p%40ssword@10.10.25.50:50005`
  Dies gilt auch für Benutzernamen.
- Nach der Anmeldung können Sie ein Ausgabeverzeichnis für die Befehle festlegen: `sdk output <some pathname>`
- Geben Sie für Hilfe zur Befehlszeile Folgendes ein: `sdk content`

Bevor Sie einige Beispielbefehle testen, müssen Sie den Parameter `sessions` verstanden haben. Dieser Parameter ist sehr wichtig und steuert, wie viele oder wie wenige Daten Sie erfassen möchten (die Where-Klausel ist ebenfalls wichtig). Beim Parameter „session“ handelt es sich entweder um eine einzelne Sitzungs-ID oder einen Bereich von Sitzungs-IDs. Alle NetWitness Suite Core-Services arbeiten mit Sitzungs-IDs, die mit 1 beginnen und für jede neue Sitzung, die dem Service hinzugefügt wird (Netzwerk- oder Protokollsitzung), um 1 erhöht werden. Sitzungs-IDs sind 64-Bit-Ganzzahlen, sodass sie sehr groß werden können. Um es einfach zu halten, wird davon ausgegangen, dass es sich um einen Log Decoder handelt, der 1.000 Protokolle aufgenommen hat und diese analysiert. Im Service sind jetzt 1.000 Sitzungen mit Sitzungs-IDs von 1 bis 1000 (die Sitzungs-ID 0 ist nie gültig) vorhanden. Wenn Sie alle 1.000 Sitzungen verarbeiten möchten, übergeben Sie „sessions=1-1000“. Wenn Sie nur die letzten 100 Sitzungen verarbeiten möchten, übergeben Sie „sessions=901-1000“. Sobald der Befehl die Verarbeitung von Sitzung 1000 abgeschlossen hat, wird wieder die Eingabeaufforderung der Konsole angezeigt.

In vielen Fällen sind jedoch keine bestimmten Sitzungsbereiche von Belang. Stattdessen soll eine Abfrage über alle Sitzungen durchgeführt werden und anschließend sollen die Sitzungen verarbeitet werden, die einer Abfrage entsprechen. Hier finden Sie einige Kürzel, die dies vereinfachen:

- Der Buchstabe `l` (Kleinbuchstabe L) steht für „untere Grenze“ oder „niedrigste Sitzungs-ID“.
- Der Buchstabe `u` steht für „höchste Sitzungs-ID“. Genau genommen steht er auch für die höchste Sitzungs-ID für zukünftige Sitzungen. Anders gesagt, wenn Sie `sessions=l-u` übergeben, bedeutet dieser spezielle Bereich, dass alle aktuellen Sitzungen im System verarbeitet werden, die Verarbeitung aber nicht beendet wird, sondern neue Sitzungen, die in das System gelangen, ebenfalls verarbeitet werden. Sobald die letzte Sitzung auf dem Service erreicht ist, wird der Befehl angehalten und es wird auf neue Sitzungen gewartet. Zusammenfassend lässt sich sagen, dass der Befehl nie beendet wird sondern in einen fortlaufenden Verarbeitungsmodus übergeht. Er wird über Tage, Monate oder Jahre ausgeführt, bis er beendet wird.
- Wenn der Befehl nicht dauerhaft ausgeführt werden soll, können Sie `now` als oberen Grenzwert übergeben. Dadurch wird die letzte Sitzungs-ID auf dem Service zum Zeitpunkt des Befehlsstarts ermittelt und alle Sitzungen werden verarbeitet, bis diese Sitzungs-ID erreicht ist. Sobald diese Sitzungs-ID erreicht ist, wird der Befehl beendet, unabhängig davon, wie viele Sitzungen seit dem Starten des Befehls zum Service hinzugefügt wurden. Für den Log Decoder im Beispiel beginnt mit `sessions=200-now` die Verarbeitung bei Sitzung 200, geht bis Sitzung 1000 und wird dann beendet. Selbst wenn dem Log Decoder nach dem Starten des Befehls weitere 1.000 Protokolle hinzugefügt werden, wird der Befehl trotzdem nach der Verarbeitung von Sitzung 1000 beendet.

- Der Parameter `sessions=now-u` bedeutet, dass die Verarbeitung bei der letzten Sitzung beginnt und alle eingehenden neuen Sitzungen verarbeitet werden. Es werden dabei keine vorhandenen Sitzungen (mit Ausnahme der letzten) verarbeitet, sondern nur neue.

Geben Sie für Beispielbefehle und deren Funktion `man sdk content examples` ein oder lesen Sie den Abschnitt Beispiele für den Befehl „sdk content“.

Beispiele für den Befehl „sdk content“

Das erste Beispiel für den NwConsole Befehl „sdk content“ unten ist einfach und zeigt alle Befehle, die Sie eingeben müssen. Die dann folgenden Beispiele zeigen nur die `sdk content`-Befehle. Im ersten Beispiel wird eine Protokolldatei erstellt und es werden die ersten 1.000 Protokolle von einem Concentrator abgerufen, der von einem Log Decoder aggregiert:

```
sdk open nw://admin:netwitness@myconcentrator.local:50005

sdk output /tmp

sdk content sessions=1-1000 render=logs append=mylogs.log
fileExt=.log
```

Dieses Skript gibt 1.000 Protokolle in die Datei **/tmp/mylogs.log** aus (vorausgesetzt, dass die Sitzungen 1 bis 1000 auf dem Service vorhanden sind). Die Protokolle liegen im Klartextformat vor. Der Parameter `fileExt=.log` ist erforderlich, um dem Befehl anzuzeigen, dass eine Protokolldatei ausgegeben werden soll.

```
sdk content sessions=1-1000 render=logs append=mylogs.log
fileExt=.log includeHeaders=true separator=","
```

Mit diesem Befehl werden dieselben 1.000 Protokolle abgerufen wie oben, es wird jedoch der Protokoll-Header analysiert und Zeitstempel, Weiterleitung und andere Informationen über das Protokoll werden extrahiert und in einer CSV-Datei abgelegt.

Beispiel CSV: 1422401778,10.250.142.64,10.25.50.66,hop04b-LC1,%MSISA-4: 81.136.243.248...

Der Zeitstempel ist in der Epoch-Zeit angegeben. Die Parameter `includeHeaders` und `separator` können nur auf NetWitness Suite-Installationen 10.4.0.2 und höher verwendet werden.

```
sdk content sessions=1-now render=logs append=mylogs.log
fileExt=.log includeHeaders=true separator=","
where="risk.info='nw35120'"
```

Dieser Befehl schreibt eine Protokolldatei über den aktuellen Sitzungsbereich, aber nur für Protokolle, die `risk.info='nw35120'` entsprechen. Denken Sie daran, dass beim Hinzufügen einer Where-Klausel angeben eine Abfrage im Hintergrund durchgeführt wird, um die Sitzungs-IDs für den Export zu erfassen. Die Abfrage sollte auf einem Service ausgeführt werden, auf dem die entsprechenden Feldern indexiert sind (normalerweise ein Broker oder Concentrator). Da Sie eine Abfrage für das Feld `risk.info` durchführen, überprüfen Sie in diesem Fall den Service, auf dem Sie den Befehl ausführen, um sicherzustellen, dass er auf Wertebene indexiert ist (IndexValues, siehe die Datei index-concentrator.xml für Beispiele). Standardmäßig ist bei den meisten Decodern nur die Zeit indexiert. Wenn Sie ein anderes Feld als das Zeitfeld in der Where-Klausel verwenden, müssen Sie die Abfrage vom Decoder auf einen Concentrator, Broker oder Archiver mit den richtigen Indexebenen für die Abfrage verschieben. Weitere Informationen zur Indexierung und zum Schreiben von Abfragen finden Sie im *Tuning-Leitfaden für NetWitness Suite-Core-Datenbanken*.

```
sdk content sessions=1-now render=logs append=mylogs.log
fileExt=.log includeHeaders=true separator="," 
where="threat.category exists && time='2015-01-05 15:00:00'-
'2015-01-05 16:00:00'"
```

Dieser Befehl entspricht dem Befehl oben, sucht aber nur nach passenden Protokollen zwischen 15:00 Uhr und 16:00 Uhr (UTC) am 5. Januar 2015, die über den Metaschlüssel `threat.category` verfügen. Da diese Abfrage ebenfalls ein anderes Feld als die Zeit in der Where-Klausel enthält (`threat.category`), sollte sie auf einem Service ausgeführt werden, auf dem `threat.category` mindestens auf Ebene von `IndexKeys` indexiert ist (die Operatoren `exists` und `!exists` erfordern nur eine Indizierung auf Schlüsselebene, obwohl Werte ebenso funktionieren).

```
sdk content sessions=1-now render=logs append=mylogs
fileExt=.log where="event.source begins 'microsoft'"
maxFileSize=1gb
```

Dieser Befehl erstellt mehrere Protokolldateien, die jeweils nicht größer als 1 GiB sind. Den Dateinamen wird **mylogs** vorangestellt und das Datum und die Uhrzeit des ersten Paket/Protokoll-Zeitstempels in der Datei angehängt. Einige Beispieldateinamen: **mylogs-1-2015-Jan-28T11_08_14.log**, **mylogs-2-2015-Jan-28T11_40_08.log** and **mylogs-3-2015-Jan-28T12_05_47.log**. Auf Versionen, die älter als Security Analytics 10.5 sind, ist das T-Trennzeichen zwischen Datum und Uhrzeit ein Leerzeichen.

```
sdk content sessions=1-now render=pcap append=mypackets
where="service=80,21 && time='2015-01-28 10:00:00'-'2015-01-28
15:00:00'" splitMinutes=5 fileExt=.pcap
```

Dieser Befehl erfasst alle Pakete im Fünf-Stunden-Zeitraum für die Servicetypen 80 und 21 und schreibt eine PCAP-Datei. Alle 5 Minuten wird eine neue PCAP-Datei geschrieben.

```
sdk content time1="2015-01-28 14:00:00" time2="2015-01-28
14:15:00" render=pcap append=mydecoder fileExt=.pcap
maxFileSize=512mb sessions=1-now
```

Seien Sie bei diesem Befehl vorsichtig. Warum? Er funktioniert für Pakete und Protokolle und ist *extrem schnell*. Der Nachteil ist, dass alles zurückgegeben wird, was zwischen den beiden Zeitbereichen liegt und Sie keine Where-Klausel verwenden können. Auch hier beginnt das Rück-Streaming nahezu sofort und es muss nicht erst eine Abfrage im Back-end durchgeführt werden. Da alles mithilfe sequenzieller I/O gelesen wird, kann die Netzwerkverbindung zwischen Server und Client voll ausgelastet werden. Es wird mit der Erstellung von Dateien begonnen, denen **mydecoder** vorangestellt wird, und mit einer neuen Datei fortgefahren, sobald eine Größe von 512 MiB erreicht ist.

```
sdk tailLogs
```

oder (der entsprechende Befehl):

```
sdk content render=pcap console=true sessions=now-u
```

Dies ist ein nützlicher Befehl. Er verwendet im Hintergrund `sdk content`. Der Zweck dieses Befehls ist es, alle eingehenden Protokolle auf einen Log Decoder anzuzeigen. Nicht mehr. Es handelt sich um einen sehr einfachen Befehl. Wenn Protokolle den Log Decoder erreichen (der Befehl kann auch auf einem Broker oder Concentrator ausgeführt werden) werden Sie auf dem Konsolenbildschirm ausgegeben. Dies ist eine hervorragende Möglichkeit, um festzustellen, ob die Erfassung auf dem Log Decoder erfolgt und was genau den Log Decoder erreicht. Dieser Befehl wird im kontinuierlichen Modus ausgeführt. Verwenden Sie ihn nicht, wenn die Erfassung auf dem Log Decoder mit der höchsten Aufnahmerate erfolgt (der Befehl kann nicht Schritt halten). Er ist aber zu Überprüfungs- oder Troubleshooting-Zwecken nützlich.

```
sdk tailLogs where="device.id='ciscoasa'"
pathname=/mydir/anotherdir/mylogs
```

Dieser Befehl ist derselbe wie oben, jedoch werden nur Protokolle ausgegeben, die der Where-Klausel entsprechen. Die Ausgabe erfolgt zudem nicht über die Konsole, sondern in einen Satz Protokolldateien unter `/mydir/anotherdir`, die nicht größer als 1 GiB werden. Dies ist natürlich auch mit dem Befehl `sdk content` möglich, aber es erfordert mit diesem Befehl weniger Schreibarbeit, wenn Ihnen das Standardverhalten ausreicht.

```
sdk content sessions=now-u render=pcap where="service=80"
append=web-traffic fileExt=.pcap maxFileSize=2gb
maxDirSize=100gb
```

Dieser Befehl beginnt mit dem Schreiben von PCAPs des gesamten Webdatenverkehrs ab der letzten Sitzung einschließlich aller neu eingehenden Sitzungen, für die „service=80“ zutrifft. Es werden PCAPs mit einer maximalen Größe von 2 GiB geschrieben und wenn alle PCAPs im Verzeichnis zusammen mehr als 100 GiB groß sind, werden so lange die ältesten PCAPs gelöscht, bis das Verzeichnis um 10 % kleiner als die maximale Größe ist. Beachten Sie, dass die Überprüfung der Verzeichnisgröße nicht präzise ist und standardmäßig nur alle 15 Minuten erfolgt. Sie können die Anzahl der Minuten zwischen den Überprüfungen durch Übergabe von `cacheMinutes` als Parameter anpassen, dies funktioniert aber nur in Security Analytics 10.5 und höher.

```
sdk content sessions=79000-79999 render=nwd
append=content-%1%.nwd metaFormatFilename=did
```

Dies ist eine Art Backupbefehl. Es werden 1.000 Sitzungen erfasst und ihr gesamter Inhalt (Sitzungen, Metadaten, Pakete oder Protokolle) wird im NWD-Format (NetWitness Data Format) ausgegeben. NWD ist ein spezielles Format, das ohne Analyse in ein Paket oder einen Log Decoder reimportiert werden kann. Prinzipiell wird die ursprünglich analysierte Sitzung ohne Änderungen importiert. Der Zeitstempel wird ebenfalls nicht verändert. Wenn die Analyse ursprünglich vor 6 Monaten erfolgte, wird beim Import der Zeitstempel mit dem Datum von vor 6 Monaten beibehalten.

Hinweis: Erwarten Sie bei diesem Befehl keine großartige Performance, insbesondere nicht bei Paketen. Das Sammeln der Pakete für eine Sitzung umfasst viel zufällige I/O und kann den Export drastisch verlangsamen. Protokolle sind von diesem Problem nicht so stark betroffen (nur ein Protokoll pro Sitzung), aber im Hintergrund dieses Befehls wird die „/sdk content"-API verwendet. Dabei handelt es sich nicht um eine performanceorientierte Streaming-API wie beispielsweise „/sdk packets". Erwarten Sie daher also keine großartige Performance.

Der Parameter `metaFormatFilename` ist bei diesem Befehl sehr hilfreich. Wenn Sie diesen Befehl auf einem Concentrator mit mehr als einem Service ausführen, werden die NWD-Dateinamen mit dem Metawert `did` für jede Sitzung erstellt („%1%" im angehängten Parameter wird durch den Wert von `did` ersetzt). Jeder Dateiname gibt genau an, von welchem Decoder die Daten stammen.

```
sdk content session=1-u where="service=80,139,25,110"
render=files maxDirSize=200mb cacheMinutes=10
```

Dies ist ein weiterer nützlicher Befehl. Er funktioniert sehr ähnlich wie unser altes Visualize-Produkt, wenn Sie das Ausgabeverzeichnis mit Windows-Explorer im Symbolmodus kombinieren. Der Befehl extrahiert die Daten aus dem gesamten Web-, E-Mail- und SMB-Datenverkehr. Dies umfasst alle möglichen Dateien, z. B. Bilder, .zip-Dateien, Videos, PDF-Dateien, Office-Dokumente, Textdateien, ausführbare Dateien und Audiodateien. Wenn dabei Schadsoftware extrahiert wird, wird diese vom Virenscanner markiert. Sie müssen sich jedoch keine Gedanken machen, durch den Befehl wird nichts ausgeführt, daher wird der Computer nicht infiziert (es sei denn, Sie versuchen selbst, die Schadsoftware auszuführen). Der Befehl kann allerdings nützlich sein, denn wenn Sie Schadsoftware finden, zeigt der Dateiname an, aus welcher Sitzungs-ID diese extrahiert wurde. Sie können dann diese Sitzungs-ID abfragen und sehen, welcher Host potenziell mit der Schadsoftware infiziert wurde und Maßnahmen ergreifen. Mit den Parametern `includeFileTypes` oder `excludeFileTypes` können Sie filtern, was extrahiert wird (siehe Hilfe zum Befehl). Wenn Sie beispielsweise `excludeFileTypes=".exe;.dmg;.msi"` hinzufügen, werden keine ausführbaren Dateien und Installationsprogramme extrahiert. Dieser Befehl wird ununterbrochenen ausgeführt und extrahiert Dateien aus allen vorhandenen und neuen Sitzungen. Sobald das Verzeichnis mehr als 200 MiB Dateien enthält, wird es automatisch alle 10 Minuten bereinigt.

Hinweis: Dieser Befehl ist nur für Paketsitzungen sinnvoll, nicht für Protokolle.

```
sdk content session=1-now where="time='2015-01-27 12:00:00'-'2015-01-27 13:00:00'
&& (service=25,110,80)" subdirFileTypes="audio=.wav;.mp3;.aac;
video=.wmv;.flv;.mp4;.mpg;.swf; documents=.doc;.xls;.pdf;.txt;.htm;.html
images=.png;.gif;.jpg;.jpeg;.bmp;.tif;.tiff archive=.zip;.rar; other=*"
renameFileTypes=".download|.octet-stream|.program|.exe;.jpeg|.jpg" render=files
maxDirSize=500mb
```

Dieser Befehl extrahiert Dateien aus HTTP- und E-Mail-Sitzungen aus einem 1-Stunden-Zeitraum und gruppiert die extrahierten Dateien dann in Verzeichnisse, die durch den Parameter `subdirFileTypes` angegeben werden. Beispielsweise werden alle extrahierten Audiodateien mit der Erweiterung .wav, .mp3 oder .aac im Unterverzeichnis „audio“ platziert, dass unter dem angegebenen Ausgabeverzeichnis erstellt wird. Gleiches gilt für alle anderen in diesem Parameter angegebenen Gruppen. Einige Dateien werden außerdem automatisch basierend auf der Dateierweiterung umbenannt. Dies wird von `renameFileTypes` ausgeführt. Jede Datei mit der Erweiterung .download, .octet-stream oder .program wird in .exe umbenannt. Dateien mit der Erweiterung .jpeg werden in .jpg umbenannt. Sobald das oberste Verzeichnis 500 MiB überschreitet, werden die ältesten Dateien bereinigt. Dieser Befehl wird bei der zum Zeitpunkt des Befehlsstarts letzten Sitzung beendet.

```
sdk search session=1-now where="service=80,25,110" search="keyword='party' sp ci"
```

Dieser Befehl durchsucht alle Pakete und Protokolle (Parameter `sp`) nach dem Schlüsselwort `party`. Wenn „party“ an einer beliebigen Stelle in den Paketen oder Protokollen gefunden wird, wird die Sitzungs-ID zusammen mit dem gefundenen Text und dem umgebenden Text als Kontext ausgegeben. Die Where-Klausel gibt an, dass nur Web- und E-Mail-Datenverkehr durchsucht wird. Der Parameter `ci` bedeutet, dass es sich um eine Suche ohne Berücksichtigung der Groß-/Kleinschreibung handelt. Sie können `regex` durch `keyword` ersetzen, dann wird eine Regex-Suche durchgeführt.

```
sdk search session=1-now search="keyword='checkpoint' sp ci" render=log
append=checkpoint-logs.log fileExt=.log
```

Dies ist ein interessantes Befehlsbeispiel. Bei diesem Befehl werden alle Protokolle (dies funktioniert auch mit Paketen) nach dem Schlüsselwort `checkpoint` durchsucht. Wird dieses gefunden, so wird das Protokoll in eine Datei **checkpoint-logs.log** extrahiert. Dieser Befehl bietet eine Vielzahl an Möglichkeiten. Prinzipiell wird die Sitzung bei einer ermittelten Übereinstimmung an den Inhaltsaufruf übergeben. Alle nicht erkannten Parameter, die Sie an `sdk search` übergeben, werden an den Inhaltsaufruf übergeben. Dadurch wird der volle Funktionsumfang des `sdk content`-Abrufs ermöglicht, aber nur für die Sitzungen ausgeführt, für die Treffer bei der Inhaltssuche erzielt werden. Leistungsstarke Befehle wollen überlegt eingesetzt werden.

Befehle für das Troubleshooting

NwConsole bietet die folgenden Befehle, die beim Troubleshooting von Security Analytics hilfreich sind:

- **whatIsWrong**: Liefert einen Snapshot von Konfiguration, Statistiken und Fehler- und Warnungsprotokollen zu einem Service für einen angegebenen vergangenen Zeitraum.
- **dbcheck**: Führt eine Konsistenzprüfung von Datenbankdateien durch
- **topQuery**: Hilft dabei, Abfragen zu ermitteln, die übermäßig viel Zeit in Anspruch nehmen
- **netbytes**: Führt ein Troubleshooting der Netzwerkverbindungen auf dem aktuellen Host durch
- **netspeed**: Führt ein Troubleshooting der Verbindung zwischen dem Hostcomputer, auf dem NwConsole ausgeführt wird, und dem mithilfe des Befehls `login` verbundenen Remotecomputer durch.

In den folgenden Abschnitten sowie auf den Seiten mit Informationen zur Hilfe und zu Themen zu NwConsole finden Sie zusätzliche Informationen.

whatIsWrong

Wenn ein Service nicht ordnungsgemäß funktioniert, ist der Grund in der Regel in den Protokollen zu finden, die der Service geschrieben hat. Sie können den Konsolenbefehl `whatIsWrong` verwenden, um einen Snapshot von Konfiguration, Statistiken und Fehler- und Warnungsprotokollen (einschließlich zugehöriger Kontextprotokolle) zu einem Service für einen angegebenen vergangenen Zeitraum abzurufen. Standardmäßig sind dies die letzten sieben Tage. Sie können die Ergebnisse des ausgeführten Befehls whatIsWrong in einer angegebenen Klartextdatei speichern. Die Ausgabe dieses Befehls kann ein hilfreicher Ausgangspunkt zur Ermittlung eines aktuellen Problems mit einem Service sein.

Um den Konsolenbefehl whatIsWrong verwenden zu können, melden Sie sich mit dem Befehl `login` bei dem Service an, für den das Troubleshooting durchgeführt werden soll, und führen Sie den Befehl `whatIsWrong` aus.

Tipp: Verwenden Sie `help whatIsWrong`, um alle verfügbaren Parameter anzuzeigen, darunter die Anzahl der einzuschließenden vergangenen Tage/Stunden, der Pfadname zum Speichern der Ergebnisse, die Entscheidung, ob die Ergebnisdatei erweitert oder überschrieben werden soll, und das für die Protokollfelder zu verwendende Trennzeichen. Sie können außerdem die Anzahl der für die Kontextermittlung zu verwendenden letzten Protokolle begrenzen und angeben, wie viele Kontextprotokolle pro Warnung/Fehler abgerufen werden sollen.

Wann immer Sie eine Anfrage für Protokolle zu einem Core-Service erhalten haben, sollten Sie zuerst den Befehl „whatIsWrong“ ausführen und die gesammelten Ergebnisse als Ausgangspunkt verwenden.

dbcheck

Der Befehl `dbcheck` dient zur Konsistenzprüfung von Datenbankdateien (Sitzung, Metadaten, Pakete, Protokolle, Statistiken usw.). Dies kann erforderlich sein, wenn ein Service aufgrund von Konsistenzfehlern in Datenbankdateien nicht gestartet werden kann. In der Regel würde ein Service automatisch wiederhergestellt und Konsistenzprobleme würden beim Start behoben werden. Es kann jedoch vorkommen, dass dies nicht erfolgt. Beim Start eines Services (z. B. Decoder) werden normalerweise die meisten Datenbankdateien nicht gelesen, um so den Start zu beschleunigen. Es wird vorausgesetzt, dass die meisten Dateien in einem konsistenten Zustand sind, und es wird nur eine flüchtige Prüfung der zuletzt geschriebenen Dateien vorgenommen. Treten Probleme auf, können diese Konsistenzprüfungen durch `dbcheck` ausgeführt werden, aber NUR, wenn der Service nicht ausgeführt wird.

Achtung: Versuchen Sie nicht, diesen Befehl auszuführen, während ein Service ausgeführt wird.

Beispielsweise können Sie eine einzelne Datei prüfen:

```
dbcheck /var/netwitness/decoder/packetdb/packet-000000001.nwpdb
```

Sie können auch Platzhalter verwenden, um mehrere Dateien zu prüfen:

```
dbcheck /var/netwitness/decoder/metadb/meta-00000002*.nwmdb
```

topQuery

Der Befehl `topQuery` kann hilfreich sein, um Abfragen zu ermitteln, die übermäßig viel Zeit in Anspruch nehmen Dieser Befehl analysiert die Auditprotokolle eines Services und gibt die Top N längsten ausgeführte Abfragen für den angegebenen Zeitraum aus.

Die einfachste Methode der Ausführung ist durch Anmeldung beim Service (in der Regel ein Broker oder Concentrator) und durch Eingabe von `topQuery`. Das Standardverhalten ist die Rückgabe der 100 am längsten ausgeführten Abfragen für die letzten sieben Tage.

Geben Sie für eine Liste der Parameter `help topQuery` ein. Hier einige weitere Beispiele mit Erklärungen:

```
topQuery hours=12 top=10
```

Dieser Befehl gibt die Top-10-Abfragen der letzten 12 Stunden zurück.

```
topQuery time1="2015-03-01 00:00:00" time2="2015-03-14 00:00:00"
```

Dieser Befehl gibt die Top-100-Abfragen zwischen dem 1. März 2015 und dem 14. März 2015 aus. Zeiten sind in UTC angegeben, nicht der lokalen Zeitzone.

```
topQuery input=/var/log/messages output=/tmp/top20.txt top=20 user=sauser1
```

Statt eine Verbindung zu einem Service herzustellen, werden die Syslog-Auditmeldungen für die Top-20-Abfragen der letzten 7 Tage analysiert, jedoch nur Abfragen, die vom Benutzer sauser1 ausgeführt wurden. Die Top-20-Abfragen werden nicht auf dem Konsolenbildschirm angezeigt, sondern in die Datei /tmp/top20.txt geschrieben. Der Parameter „user“ ist ein Regex, sodass Sie mehrere Benutzernamen angeben können, indem Sie etwa schreiben: `user="(sauser1|sauser2)"`.

netbytes

Der Befehl `netbytes` ist sehr nützlich beim Troubleshooting der Netzwerkverbindungen auf dem aktuellen Host. Er zeigt kontinuierliche Statistiken zum Senden und Empfangen aller Netzwerkschnittstellen an. Nach der Ausführung müssen Sie **STRG+C** drücken, um diesen Befehl zu beenden, wodurch auch die NwConsole beendet wird.

netspeed

Der Befehl `netspeed` dient dazu, ein Troubleshooting der Verbindung zwischen dem Hostcomputer, auf dem NwConsole ausgeführt wird, und dem mithilfe des Befehls `login` verbundenen Remotecomputer durchzuführen. Sie geben die Anzahl der zu übertragenden Byte an und der Befehl misst die Geschwindigkeit der Verbindung. Der Befehl „netspeed“ ist sehr nützlich für das Troubleshooting von Problemen mit der Aggregationsperformance, die mit dem Netzwerk zusammenhängen könnten.

```
login somedecoder:50004 admin ...

netspeed transfer=4g
```

Stellen Sie für das Troubleshooting der Verbindung zwischen einem Concentrator und einem Decoder über SSH eine Verbindung zum Concentrator her, starten Sie NwConsole und melden Sie sich dann bei dem Decoder an und führen Sie `netspeed` aus. Die Ausgabe durch den Befehl zeigt Ihnen den maximalen Netzwerkdurchsatz an. Wenn dieser deutlich geringer als die standardmäßige 1-Gbit/s-Schnittstelle ist, kann dies auf ein Netzwerkproblem hinweisen.