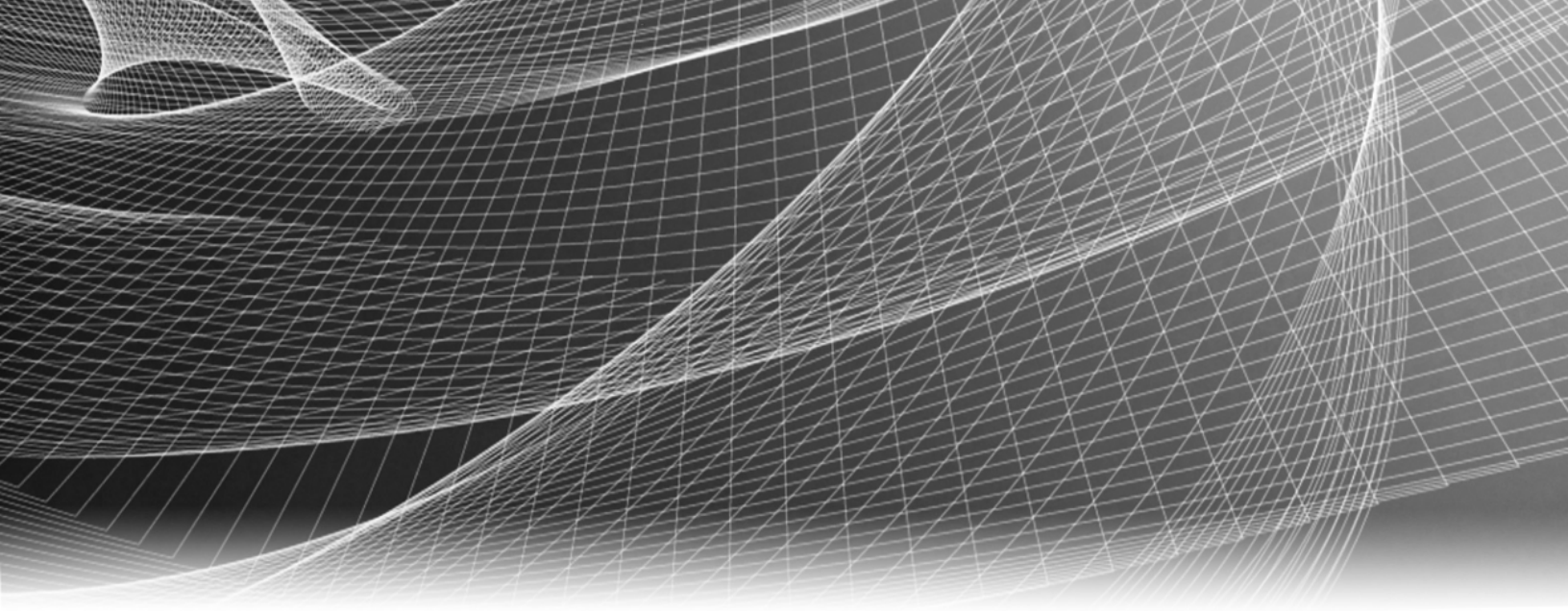

RSA Security Analytics

Interface de ligne de commande de Security Analytics pour la Version 10.6

RSA

EMC²

Trademarks

License Agreement

Third-Party Licenses

This product may include software developed by parties other than RSA. The text of the license agreements applicable to third-party software in this product may be viewed in the thirdpartylicenses.pdf file.

Note on Encryption Technologies

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when using, importing or exporting this product.

Distribution

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license. EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED "AS IS." EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Interface de ligne de commande de Security Analytics

Interface de ligne de commande de Security Analytics

Security Analytics présente les informations dans un navigateur Web sous forme de tableaux de bord et de vues. La plupart des utilisateurs configurent et utilisent Security Analytics à l'aide de l'interface utilisateur, mais certains utilisateurs avancés, tels que les administrateurs et les développeurs, peuvent avoir des tâches qui nécessitent un accès de la ligne de commande à Security Analytics. La console RSA Security Analytics, également appelée NwConsole, est un utilitaire multiplates-formes qui fournit les outils et l'accès de la ligne de commande aux services Security Analytics Core dont les utilisateurs Security Analytics ont besoin.

Last Modified: August 10 2016, 9:33AM

Console RSA Security Analytics

La console RSA Security Analytics, également appelée NwConsole, est une application de terminal multiplates-formes qui fournit des outils puissants et un accès de la ligne de commande aux services Security Analytics Core, tels que Decoder, Log Decoder, Concentrator, Broker et Archiver. La plupart des utilisateurs effectuent leurs tâches et procédures d'enquête via l'interface utilisateur de Security Analytics, mais certains utilisateurs avancés, tels que les administrateurs et les développeurs, ont besoin d'un accès direct aux services Security Analytics Core sans passer par l'interface utilisateur. NwConsole vous permet de saisir des commandes à partir de la ligne de commande ou d'exécuter plusieurs commandes à partir d'un fichier.

L'application Console RSA Security Analytics est installée sur toutes les appliances Security Analytics. Vous pouvez également l'installer sur Windows, Mac et CentOS pour vous connecter à un service Core et interagir avec lui. Pour obtenir le programme d'installation de l'application Console RSA Security Analytics, contactez le Support Clients RSA.

Last Modified: August 10 2016, 9:33AM

Accéder à NwConsole et aide

Cette rubrique décrit comment accéder à NwConsole et afficher l'aide interne à NwConsole.

Les informations d'aide détaillées sont disponibles dans la console RSA Security Analytics, également connue sous le nom de NwConsole. Vous pouvez effectuer cette opération à partir de la ligne de commande Security Analytics.

Conditions préalables

NwConsole est disponible à partir de la ligne de commande sur une appliance Security Analytics. Si vous accédez à une appliance principale à distance, l'application console RSA Security Analytics doit être installée sur une machine Windows, Mac ou CentOS. Pour obtenir le programme d'installation de l'application console RSA Security Analytics, contactez le Support Clients RSA.

Accéder à NwConsole

Pour exécuter NwConsole à partir de la ligne de commande sur une appliance Security Analytics ou un émulateur de terminal, à l'invite <$>, saisissez `NwConsole` (Linux) ou `nwconsole` (Windows). La commande réelle est `NwConsole`, mais Windows n'est pas sensible à la casse. La console RSA Security Analytics s'affiche comme illustré dans l'exemple suivant.

```
Last login: Thu Sep 24 14:00:42 on console
usxx<username>m1:~ <username>$ NwConsole
RSA Security Analytics 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.
>
```

Afficher l'aide

NwConsole fournit de l'aide pour chaque commande, mais aussi de l'aide pour des rubriques spécifiques.

⚠ **Caution:** Pour obtenir les dernières informations disponibles, consultez les rubriques d'aide et les rubriques relatives aux commandes au sein de NwConsole.

Last Modified: August 10 2016, 9:33AM

Afficher la liste des commandes

Pour afficher la liste des commandes disponibles et leurs descriptions, à l'invite (>), saisissez `help`. L'exemple suivant affiche la liste des commandes disponibles.

```
> 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
```


Last Modified: August 10 2016, 9:33AM

```
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>"
>
```

Afficher l'aide détaillée d'une commande

Pour afficher les informations détaillées relatives à une commande, saisissez `help <commande>`. L'exemple suivant montre l'aide relative à la commande `logParse` après avoir saisi `help logParse`.

```
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
>
```

Afficher la liste des rubriques d'aide

Pour afficher la liste des rubriques d'aide, saisissez `man`. L'exemple suivant affiche la liste des rubriques d'aide.

```
> 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
>
```

Afficher une rubrique d'aide spécifique

Pour afficher l'aide relative à une rubrique spécifique, saisissez `man <rubrique>`. L'exemple suivant affiche la rubrique d'aide Paquets après avoir saisi `man Packets`.

```
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.
>
```


Last Modified: August 10 2016, 9:33AM

⚠ **Caution:** Pour obtenir les dernières informations disponibles, consultez les rubriques d'aide et les rubriques relatives aux commandes au sein de NwConsole.

Quitter NwConsole

Pour quitter l'application NwConsole, saisissez `quit` dans la ligne de commande.

Last Modified: August 10 2016, 9:33AM

RSA

Paramètres de base et modification de la ligne de commande

NwConsole ressemble à un couteau suisse. Il existe toutes sortes d'outils derrière son interface de ligne de commande. NwConsole est utilisable sur plusieurs platesformes. Des exécutables sont disponibles pour CentOS (déjà disponible sur les appliances), Windows et Mac.

Paramètres de ligne de commande de base

Voici quelques paramètres de ligne de commande de base :

- Pour exécuter un ensemble de commandes à partir d'un fichier :
  `NwConsole f /tmp/somefile.script`
- Pour passer une liste de commandes à partir de la ligne de commande :
  `NwConsole c <command1> c <command2> c <command3>`
  Ceci n'est pas nécessairement recommandé sauf pour les scripts très simples. L'interpréteur bash risque de mal gérer les chaînes avec des guillemets si vous n'utilisez pas la séquence d'échappement appropriée. Si vous faites passer des erreurs non évidentes via la ligne de commande, essayez plutôt la lecture d'un fichier pour voir si cela résout les problèmes.
- Normalement, la console se ferme après l'exécution des commandes passées via un fichier ou une ligne de commande. Toutefois, si vous voulez maintenir l'invite ouverte de manière interactive après l'exécution des commandes, passez `i` sur la ligne de commande.
- Naturellement, vous pouvez tout simplement exécuter NwConsole et entrer les commandes dans la fenêtre de la console.

Last Modified: August 10 2016, 9:33AM

Modification de la ligne de commande

Vous pouvez utiliser les touches du tableau suivant lors de la modification d'une commande.

Clé	Description
CtrlU	Efface la ligne en cours
CtrlW	Supprime le mot sur lequel se trouve le curseur
CtrlA	Déplace le curseur au début de la ligne
CtrlE	Déplace le curseur à la fin de la ligne
Flèche vers le haut	Affiche la commande précédemment exécutée
Flèche vers le bas	Affiche la commande exécutée après la commande en cours (valable uniquement si vous avez appuyé sur la touche Flèche vers le haut)
Flèche gauche	Déplace le curseur vers le caractère précédent
Flèche droite	Déplace le curseur vers le caractère suivant
Tab	Achève le remplissage contextuel de la plupart des commandes et leurs paramètres. La touche Tab est très utile pour les modifications. Par exemple, pour afficher la rubrique d'aide *Connexion à un service*, placezvous sur la ligne de commande, entrez `man con`, puis appuyez sur la touche `Tab`. NwConsole finit la commande à votre place : `man Connexion à un service` Appuyez sur Entrée pour exécuter la commande et visualiser la rubrique.
`historique`	Affiche une liste numérotée des commandes précédentes
`history execute=#`	Exécute une commande précédente, ce qui équivaut à entrer `!#` Par exemple, `!1` exécute la commande précédente.
`history clear`	Efface l'intégralité de l'historique des commandes
`history erase=#`	Efface une commande spécifique dans la mémoire tampon de l'historique. L'historique est automatiquement stocké d'une session à l'autre.

Last Modified: August 10 2016, 9:33AM

RSA

Connexion à un service

Pour se connecter et interagir avec un service Security Analytics Core (Decoder, Concentrator, Broker, Archiver, etc.), vous devez émettre au préalable la commande `login`. Vous devez disposer d'un compte sur ce service. À tout moment, vous pouvez vous connecter en mode `type help` pour plus d'informations. Voici la syntaxe de la commande `login` :

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

Par exemple : **login 10.10.1.15:56005:ssl someuser**

Si vous ne spécifiez pas de mot de passe, vous serez invité(e) à le saisir ; le mot de passe apparaîtra masqué.

Si vous avez configuré la confiance appropriée entre NwConsole et le point de terminaison, vous pourrez utiliser la commande `tlogin` et éviter d'avoir à saisir un mot de passe. La configuration de la confiance n'entre pas dans le cadre de cette documentation, mais elle consiste à ajouter le certificat SSL de NwConsole au point de terminaison via la commande `send /sys peerCert op=add file-data=<nomchemin du certificat>`. Vous devez d'abord utiliser une connexion normale avec les autorisations appropriées avant de pouvoir ajouter un certificat homologue pour les connexions de confiance suivantes.

Une fois connecté(e), vous pouvez interagir avec le service du point de terminaison via un système de fichiers virtuel. Au lieu de rechercher des fichiers, vous devez rechercher les nœuds de ce service. Certains nœuds correspondent à des dossiers et disposent de nœuds enfants constituant une structure hiérarchique. Chaque nœud a une utilité et tous prennent en charge un sous-ensemble de commandes telles que `info` et `help`. Le message `help` renvoie des informations sur les commandes que chaque nœud prend en charge. Lorsque vous vous connectez pour la première fois, vous vous trouvez sur le nœud, qui correspond au chemin d'accès `/`, comme pour un système Linux ou Mac. Pour afficher une liste de nœuds sous `/`, saisissez la commande `ls`.

Tous les services disposent de nœuds tels que `sys` et `logs`. Pour interagir avec l'API **/logs**, vous pouvez dans un premier temps envoyer la commande `help` au nœud **/logs**. Pour cela, vous devez utiliser le message `send` avec cette syntaxe :

```
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
```

Donc, pour envoyer un message `help`, vous allez envoyer ce qui suit :

```
send /logs help
```

Et votre réponse doit ressembler à ce qui suit :

```
   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>]
```

Pour obtenir plus d'informations sur un message ou une commande spécifique, vous pouvez spécifier `msg=<nom message>` au sein de la commande d'aide en guise de paramètre. Par exemple, recherchez l'aide du message `pull` :

```
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)
```

L'aide du message intégrée indique que ce message capture les dernières entrées de log N si vous abandonnez id1 et id2. Pour consulter les 10 dernières entrées de log de ce service, saisissez :

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

Presque toutes les commandes du service applique ce format simple. Les seules commandes qui ne procèdent pas ainsi sont celles qui requièrent un protocole compliqué, comme l'importation d'un PCAP vers un service Decoder. Pour importer un PCAP, utilisez la commande `import` de NwConsole, qui prend en charge le protocole de canal de communication compliqué.

Certains paramètres sont spécifiques à la commande `send` de NwConsole et en réalité ne sont pas envoyés au service. Vous pouvez utiliser ces paramètres pour modifier le format de sortie de la réponse, écrire la réponse dans un fichier, ou lire un fichier à partir de la machine locale et l'envoyer au service.

- `output-format` — Ce paramètre remplace la sortie normale de la commande au format texte brut par l'un de ces types : JSON, XML ou HTML.
- `output-pathname` — Au lieu d'écrire le résultat sur le terminal, il l'écrit dans le dossier spécifié (tout fichier existant est tronqué).
- `output-append-pathname` — Comparable à `output-pathname` sauf que la sortie est ajoutée à un fichier existant (ou un fichier est créé s'il n'existe pas).
- `file-data` — Lit un fichier et l'utilise en tant que charge utile de commande. Cela est utile pour les commandes telles que `/sys fileEdit`. L'exemple suivant indique comment envoyer un fichier **index-concentrator-custom.xml** mis à jour à l'aide de NwConsole :

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

- `string-data` — Envoie la charge utile de commande en tant que chaîne au lieu d'une liste de paramètres.
- `binary-data` — Envoie la charge utile de commande en tant que binaire au lieu d'une liste de paramètres.

Exemple de requête de flux dans un fichier JSON (peut correspondre à un ensemble de résultats importants) :

```
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
```

Pour parcourir la hiérarchie des nœuds virtuels du service, vous pouvez utiliser la commande `cd` de la même manière que la commande shell. Cela couvre les bases en matière de connexion et d'interaction avec un service. Une fois que vous êtes connecté(e), la commande `help` affiche toutes les commandes que vous pouvez utiliser pour interagir avec le point de terminaison. Si vous n'êtes pas connecté(e) à un point de terminaison, ces commandes ne s'affichent pas.

Statistiques de surveillance

Vous pouvez utiliser NwConsole pour observer les statistiques d'un service en temps réel. Cependant, il faut savoir que cela peut générer un nombre IMPORTANT de résultats. Si vous ne faites pas attention, vous risquez de suivre trop de nœuds ; résultat : l'affichage défile trop rapidement pour être exploitable.

Par exemple, si vous vous connectez à un service Decoder, vous pouvez surveiller le taux de capture en temps réel. Pour ce faire, émettez ces commandes après la connexion à un service Decoder :

```
decoder/stats
```

```
mon capture.rate
```

Voilà, c'est tout ce que vous devez faire ! Désormais, chaque fois que le taux de capture change, il s'affichera dans la fenêtre de la console.

Vous pouvez ajouter un autre moniteur :

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

Maintenant, ces deux statistiques sont observées et ces valeurs sont renvoyées lors des changements. Vous avez peut-être remarqué qu'en tentant de saisir la deuxième commande, la sortie du moniteur d'origine a désorganisé votre affichage. Il s'agit du problème connu lié aux statistiques de surveillance. Ce service a été conçu principalement pour observer les statistiques après la saisie de la première commande.

Cependant, vous pouvez arrêter le suivi en saisissant la commande `delmons`, puis en appuyant sur la touche **Entrée**. Ignorez les résultats lors de la saisie, vous serez renvoyé(e) à l'invite de commandes appropriée. Si vous souhaitez surveiller plusieurs statistiques à la fois, il vous suffit de fournir le chemin d'accès au dossier stat parent pour que toutes les statistiques figurant sous ce dossier soient surveillées. Par exemple, la saisie de `mon /decoder/stats` ou `mon`. (équivalences) permet de tout surveiller. Attendez-vous à un grand nombre de résultats ! Pensez à saisir la commande `delmons` en cas de défilement trop rapide de l'affichage.

Last Modified: August 10 2016, 9:33AM

RSA

Commandes utiles

Les commandes de la console NwConsole suivantes sont utiles lors de l'interaction avec les services de base Security Analytics :

- **feed** : Permet de créer et d'utiliser les fichiers de feed.
- **makepcap** : Convertit les fichiers DB (Packet database) en fichiers PCAP.
- **packets** : Récupère les paquets ou les logs auprès du service connecté.
- **hash** : Crée ou vérifie les hachages des fichiers de base de données.

Les sections suivantes, ainsi que l'aide de la console NwConsole et les pages d'informations sur les rubriques (man), fournissent des informations supplémentaires.

Flux

La commande `feed` fournit plusieurs utilitaires pour la création et l'examen des fichiers de feed. Un fichier de feed contient la définition et les données d'un seul feed dans un format qui a été précompilé pour un chargement efficace par un service Decoder ou Log Decoder. Pour une référence complète sur les définitions des feeds, reportez-vous à la rubrique *Fichier de définition de feed* accessible sur le site sadocs.emc.com/fr-fr.

create

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

La commande `feed create` génère des fichiers de feed pour chaque feed défini dans un fichier de définition de feed. Un fichier de définition est un document XML qui contient une ou plusieurs définitions. Chaque définition de feed spécifie un fichier de données et la structure de ce fichier de données. Les fichiers de feed résultants sont créés dans le même répertoire que le fichier de définition avec le même nom que le fichier de données, mais avec l'extension remplacée par **.feed** (par exemple, les résultats **datafile.csv** dans **datafile.feed**). Tous les fichiers existants portant le nom cible sont écrasés sans notification.

```
$ ls
   example-definition.xml    example-data.csv
   $ NwConsole
   RSA Security Analytics 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
$
```

(Facultatif) Les fichiers de feed peuvent être obfusqués en utilisant l'option `-x` suivie d'un mot de passe d'au moins 16 caractères (sans espace). Il est appliqué à tous les feeds définis dans le fichier de définition. En plus du fichier de feed, un fichier token est généré pour chaque fichier de feed. Le fichier token doit être déployé avec le fichier de feed correspondant.

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

stats

```
feed stats <feedfile>
```

La commande `feed stats` fournit des informations récapitulatives sur un fichier de feed existant, non obfusqué. La spécification d'un fichier de feed obfusqué se traduit par une erreur.

```
> 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>
```

La commande `feed dump` génère une paire de clé-valeur normalisée affichant un fichier de feed non obfusqué. Vous pouvez utiliser le fichier résultat pour valider un fichier de feed ou aider à déterminer les enregistrements considérés comme non valides lors de la création du feed. La spécification d'un fichier de feed obfusqué se traduit par une erreur. Si `outfile` est présent, la commande s'annule sans écraser le fichier existant.

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

Conversion des fichiers Packet DB en fichiers PCAP

Vous pouvez utiliser la commande `makepcap` pour convertir rapidement un fichier Packet DB en un fichier PCAP générique, en conservant l'ordre horaire des captures. Cette commande offre de nombreuses options (voir `help makepcap`), mais elle est facile à utiliser. Le répertoire Packet DB est indispensable pour le démarrage de cette commande (via le paramètre `source=<pathname>`).

Note: Vous devez arrêter le service Decoder ou Archiver avant d'exécuter cette commande. Si vous souhaitez générer un fichier PCAP lors de l'exécution du service, utilisez la commande `packets`.

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

Cette commande convertit chaque fichier Packet DB en un fichier PCAP correspondant dans le même répertoire. Si le disque est presque plein, voir la commande suivante.

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

Cette commande écrit tous les fichiers PCAP de sortie dans le répertoire figurant sous **/media/usb/sde1**.

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

Cette commande convertit uniquement les fichiers numérotés entre 4 et 6, et ignore tous les autres fichiers. En d'autres termes, il convertit les fichiers Packet DB : **packet-000000004.nwpdb**, **packet-000000005.nwpdb** et **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
```

Cette commande extrait uniquement les paquets avec un horodatage compris entre le 1er mars 2015 à 14:00 et le 2 mars 2015 à 07h30. Elle écrit le fichier au format pcapng dans le même répertoire que la source. Tous les horodatages sont au format UTC.

Paquets

Vous pouvez utiliser la commande `packets` pour générer un fichier PCAP ou log basé sur une liste d'ID de session, une période de temps, ou une clause Where. Cette commande est très flexible, vous pouvez l'utiliser avec n'importe quel service en cours d'exécution ayant accès aux données brutes d'un composant en aval. Avant d'exécuter la commande, vous devez d'abord vous `connecter` à un service, puis changer de répertoire pour accéder au nœud sdk approprié (par exemple, `cd /sdk`). Contrairement à la commande `makepcap`, qui ne fonctionne que sur le système de fichiers local, vous pouvez utiliser cette commande pour un service à distance.

```
login ...
```


Last Modified: August 10 2016, 9:33AM

```
cd /sdk

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

Cette commande consigne 10 minutes de données de paquets uniquement HTTP à partir du 1er mars dans le fichier **/tmp/march-1.pcap**. Toutes les heures sont au format UTC.

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

Cette commande consigne tous les paquets en deux fois dans un fichier compressé GZIP sous **/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
```

Cette commande écrit tous les logs en deux fois dans un fichier brut sous **/media/sdd1/mylogs.log**. Tout chemin d'accès se terminant par **.log** indique que le format du fichier de sortie doit être un log brut délimité par une ligne.

Vérification des hachages de base de données

Par défaut, Archiver crée un fichier XML pour chaque fichier DB qui est consigné. Ce fichier XML se termine par l'extension **.hash** et contient un hachage du fichier, ainsi que d'autres informations pertinentes. Vous pouvez utiliser la commande `hash` pour vérifier que le fichier DB n'a pas été altéré par la lecture de la table de hachage stockée dans le fichier XML. Vous pouvez ensuite hacher à nouveau le fichier DB pour vérifier que le hachage est valide.

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

Cette commande vérifie que le fichier Packet DB **packet-000004880.nwpdb** correspond toujours au hachage dans le fichier XML **packet-000004880.nwpdb.hash**. Pour une sécurité optimale, le fichier de hachage doit être stocké à un autre emplacement pour éviter l'altération du fichier XML (comme pour les médias non réinscriptibles), mais la commande de hachage elle-même ne tient pas compte de son lieu de stockage.

Commande SDK Content

L'une des puissantes commandes de NwConsole est `sdk content`. Elle propose de nombreuses options pour l'extraction de contenu de la pile Security Analytics Core. Vous pouvez l'utiliser pour créer des fichiers PCAP et des fichiers log, ou extraire des fichiers de sessions réseau (par exemple, pour extraire toutes les images de sessions d'e-mails). Elle permet d'ajouter des fichiers, d'attribuer une taille maximale avant de créer un nouveau fichier et de nettoyer automatiquement des fichiers lorsque le répertoire devient trop volumineux. Elle exécute des requêtes en arrière-plan pour trouver de nouvelles sessions. Elle sépare les requêtes en groupes faciles à gérer et réalise ces opérations de manière automatique. Lorsque le groupe est épuisé, elle lance une nouvelle requête pour obtenir un nouvel ensemble de données pour effectuer d'autres opérations. La liste d'options disponibles pour la commande sdk content est particulièrement étendue.

Du fait que la commande propose de si nombreuses options, ce document donne des exemples de commandes pour différents exemples d'utilisation.

Avant de pouvoir exécuter `sdk content`, vous devez exécuter différentes commandes (comme la connexion à un service). Voici quelques exemples :

- Connectez-vous d'abord à un service :
  `sdk open nw://admin:netwitness@10.10.25.50:50005`
- Si vous souhaitez vous connecter via SSL, utilisez le protocole nws :
  `sdk open nws://admin:netwitness@10.10.25.50:56005`
- Gardez à l'esprit que si vous passez une URL, vous devez chiffrer l'URL correctement. Si le mot de passe est `p@ssword`, l'URL ressemble à cela : `sdk open nw://admin:p%40ssword@10.10.25.50:50005`
  Il en va de même pour le nom d'utilisateur.
- Lorsque vous êtes connecté, vous pouvez définir un répertoire de sortie pour les commandes : `sdk output <nom de chemin d'accès>`
- Pour obtenir de l'aide sur la ligne de commande, saisissez : `sdk content`

Avant de tester des exemples de commandes, il est important de comprendre le paramètre `sessions`. Ce paramètre est particulièrement important et contrôle la quantité de données que vous souhaitez extraire (la clause where est également importante). Le paramètre sessions est un ID de session unique ou une plage d'ID de sessions. Tous les services Security Analytics Core utilisent les ID de sessions qui commencent par 1 et augmentent d'1 à chaque nouvelle session ajoutée au service (session de log ou réseau). Les ID de session sont des nombres entiers de 64 bits, donc ils sont assez longs. Pour simplifier, considérons que Log Decoder a intégré et analysé 1 000 logs. Dans le service, vous avez maintenant 1 000 sessions avec des ID de sessions entre 1 et 1 000 (l'ID de session 0 n'est jamais valide). Si vous souhaitez opérer l'intégralité des 1 000 sessions, passez la commande sessions=1-1000. Si vous ne souhaitez opérer que les 100 dernières sessions, passez la commande sessions=901-1000. Lorsque la commande termine le traitement de la 1000e session, elle revient à l'invite de commande.

Last Modified: August 10 2016, 9:33AM

Souvent, les plages de sessions spécifiques n'ont pas d'intérêt. Nous voulons seulement exécuter une requête sur toutes les sessions et traiter les sessions qui correspondent à la requête. Voici quelques raccourcis pour simplifier cette procédure :

- La lettre `l` (L minuscule) indique la limite inférieure ou l'ID de session le plus bas.
- La lettre `u` indique l'ID de session le plus élevé. D'ailleurs, il indique l'ID de session le plus élevé également pour les sessions futures. En d'autres mots, si vous passez la commande `sessions=l-u`, cette plage particulière indique d'opérer sur toutes les sessions actuelles du système, mais également de ne pas arrêter le traitement, et de traiter les nouvelles sessions au fur et à mesure qu'elles entrent sur le système. La commande se met en pause et attend de nouvelles sessions lorsque la dernière session du service est atteinte. Pour résumer, la commande ne s'arrête jamais et passe en mode de traitement continu. Elle s'exécute pendant des jours, des mois ou des années, à moins qu'elle ne soit arrêtée de force.
- Si vous ne souhaitez pas que la commande s'exécute sans fin, vous pouvez passer la commande `now` en tant que limite supérieure. Elle détermine le dernier ID de session du service au moment où la commande se lance et traite toutes les sessions jusqu'à ce que cet ID soit atteint. Lorsque cet ID de session est atteint, la commande s'arrête, quelle que soit la quantité de sessions ajoutée depuis que la commande est lancée. Ainsi, pour l'exemple du Log Decoder, `sessions=200-now` lance le traitement à la session 200, jusqu'à la session 1000, puis s'arrête. Même si 1 000 logs ont été ajoutés au Log Decoder après le début de la commande, la commande s'arrête après le traitement de la session 1 000.
- Le paramètre `sessions=now-u` indique de commencer le traitement à la toute dernière session, puis de continuer le traitement de toutes les nouvelles sessions entrantes. Aucune session existante n'est traitée (à l'exception de la dernière) : seules les nouvelles sessions le sont.

Pour obtenir des exemples de commandes et leur description, saisissez `man sdk content examples` ou consultez la rubrique Exemples de commandes SDK Content.

RSA

Exemples de commandes SDK content

Le premier exemple de commande sdk content NwConsole ci-dessous est simple et présente toutes les commandes que vous devez saisir. Ensuite, les exemples ne contiennent que les commandes `sdk content`. Le premier exemple crée un fichier log et extrait les 1 000 premiers logs d'un Concentrator qui agrège les données issues d'un Log Decoder :

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

sdk output /tmp

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

Ce script génère 1 000 logs (à supposer que les sessions 1 à 1 000 existent sur le service) dans le fichier **/tmp/mylogs.log**. Les logs sont en texte brut. Le paramètre `fileExt=.log` est nécessaire pour indiquer à la commande que nous souhaitons générer un fichier log.

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

Cette commande extrait les mêmes 1 000 logs que ci-dessus, mais elle analyse l'en-tête de log et extrait l'horodatage du log, le redirecteur et d'autres informations, et les place dans un fichier au format CSV.

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

L'horodatage figure en heure Epoch. Les paramètres `includeHeaders` et `separator` ne peuvent être utilisés que sur les installations Security Analytics 10.4.0.2 et versions supérieures.

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

Cette commande écrit un fichier log dans l'ensemble des sessions actuelles, mais seuls les logs qui correspondent à `risk.info='nw35120'`. N'oubliez pas que, lorsque vous ajoutez une clause where, elle exécute une requête en arrière-plan pour collecter les ID de session en vue de l'exportation. La requête doit être exécutée sur un service avec les champs appropriés indexés (en général un Broker ou un Concentrator). Dans ce cas, étant donné que vous interrogez le champ `risk.info`, vérifiez le service dans lequel vous exécutez la commande pour vous assurer qu'il est indexé au niveau de la valeur (Pour le paramètre IndexValues, reportez-vous au fichier index-concentrator.xml pour obtenir des exemples). Par défaut, la plupart des Decoder n'ont que time indexé. Si vous utilisez tous les champs sauf time dans la clause where, vous devez déplacer la requête du Decoder vers un Concentrator, Broker ou Archiver. Pour

plus d'informations sur l'indexation et l'écriture des requêtes, reportez-vous au Guide d'optimisation de la base de données principale de Security Analytics.

```
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'"
```

Cette commande est identique à la commande ci-dessus,mais elle ne recherche que les logs correspondant entre 15h00 et 16h00 (UTC) le 5 janvier 2015 qui ont une clé de méta `threat.category`. Encore une fois, étant donné que cette requête contient un champ autre que time dans la clause where (`threat.category`), elle doit être exécutée sur un service avec la clé de méta `threat.category` indexée, au moins au niveau `IndexKeys` (les opérateurs `exists` et `!exists` n'ont besoin que d'un index au niveau de la clé, bien que des valeurs puissent être acceptées aussi).

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

Cette commande crée différents fichiers log, chacun ne dépassant pas 1 Gio. Elle fait précéder les noms de fichier du préfixe **mylogs** et leur ajoute la date et l'heure du premier horodatage de paquet/log figurant dans le fichier. Exemples de noms de fichier : **mylogs-1-2015-Jan-28T11_08_14.log**, **mylogs-2-2015-Jan-28T11_40_08.log** et **mylogs-3-2015-Jan-28T12_05_47.log**. Pour les versions antérieures à Security Analytics 10.5, le séparateur T entre la date et l'heure est un espace.

```
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
```

Cette commande extrait tous les paquets pour la période de cinq heures pour les types de services 80 et 21 et enregistre un fichier PCAP. Elle démarre un fichier PCAP toutes les cinq minutes.

```
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
```

Prêtez attention à cette commande. Pourquoi ? Elle fonctionne aussi bien pour les paquets que pour les logs et est *extrêmement rapide*. Son inconvénient est que vous obtenez tout entre les deux périodes et que vous ne pouvez pas utiliser de clause where. Une fois encore, elle renvoie tous les résultats presque immédiatement et n'exige pas d'exécution de requête préalable sur le backend. Étant donné que tout est lu à l'aide d'E/S séquentielles, elle peut complètement saturer la liaison réseau entre le serveur et le client. Elle crée des fichiers avec le préfixe **mydecoder** et commence un nouveau fichier lorsque la taille du fichier atteint 512 Mio.

```
sdk tailLogs
```

ou (la commande équivalente) :

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

C'est une petite commande amusante. En fait, elle utilise `sdk content` en arrière-plan. Cette commande vise à afficher tous les logs entrants sur un Log Decoder. Voilà. C'est très simple. Lorsque les logs arrivent dans le Log Decoder (vous pouvez l'exécuter sur un Broker ou un Concentrator aussi), ils sortent sur l'écran de la console. C'est un excellent moyen de savoir si le Log Decoder effectue une capture et de repérer exactement ce qui entre dans celui-ci. Cette commande s'exécute en mode continu. Évitez de l'utiliser si le Log Decoder effectue une capture à un rythme d'acquisition élevé (cette commande ne peut pas suivre son rythme). Cependant, elle est très utile pour vérifier ou résoudre les problèmes.

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

Cette commande est identique à la commande ci-dessus, sauf qu'elle ne génère que les logs qui correspondent à la clause where et que, au lieu d'afficher ses résultats sur la console, elle les enregistre dans une série de fichiers dans le répertoire `/mydir/anotherdir` qui ne dépasse pas 1 Gio. Vous pouvez évidemment effectuer aussi cela avec la commande `sdk content`, mais la présente commande demande moins d'opérations de saisie.

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

Cette commande écrit les fichiers PCAP de l'ensemble du trafic Web issu de la dernière session et de toutes les nouvelles sessions entrantes qui correspondent à service=80. Elle génère des PCAP ne dépassant pas 2 Gio et, si tous les PCAP du répertoire dépassent 100 Gio, elle supprime les PCAP les plus anciens jusqu'à ce que le répertoire soit de 10 % inférieur à la taille maximale. N'oubliez pas que la vérification de la taille du répertoire n'est pas exacte et que la taille du répertoire n'est vérifiée que toutes les 15 minutes par défaut. Vous pouvez ajuster l'intervalle en minutes entre les vérifications en passant le paramètre `cacheMinutes`, mais cela ne fonctionne qu'avec Security Analytics 10.5 et versions supérieures.

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

C'est une commande de sauvegarde de pauvre. Elle extrait 1000 sessions et sort le contenu intégral (sessions, méta, paquets ou logs) au format NWD (NetWitness Data Format). NWD est un format spécial qui peut être réimporté dans un Packet Decoder ou un Log Decoder sans réanalyser. Ainsi, la session analysée d'origine importe essentiellement sans modifications. L'horodatage ne change pas aussi, et si elle a été analysée initialement il y a 6 mois, l'horodatage effectué à l'importation sera conservé tel qu'il était il y a six mois.

> **Note:** Ne vous attendez pas à obtenir de grandes performances avec cette commande, en particulier avec les paquets. La collecte des paquets pour une session implique beaucoup d'E/S aléatoires et peut ralentir drastiquement l'exportation. Les logs ne souffrent pas autant de ce problème (un seul log par session), mais cette commande utilise en arrière-plan l'API de contenu /sdk, qui n'est pas une Streaming API performante comme /sdk packets. Donc ne vous attendez pas encore à obtenir de grandes performances.

Le paramètre `metaFormatFilename` est très utile dans cette commande. Si cette commande est exécutée sur un Concentrator avec plusieurs services, les noms de fichier NWD sont créés avec le méta `did` pour chaque session (le

%1% du paramètre append est remplacé par la valeur de `did`). Chaque nom de fichier indiquera exactement le Decoder d'où proviennent les données.

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

Voilà une autre petite commande amusante. Elle fonctionne de manière très similaire à notre ancien produit Visualize si vous appariez le répertoire de sortie à Windows Explorer en mode Icône, par exemple. Elle extrait les fichiers de l'ensemble du trafic Web, de messagerie et SMB. Sont compris tous les types de fichiers (images, fichiers zip, vidéos, PDF, documents Office, fichiers texte, exécutables et fichiers audio. Si elle extrait un malware, votre analyseur de virus le repérera. Ne vous inquiétez pas, rien ne sera exécuté par la commande et la machine ne sera pas infectée (sauf si vous tentez d'exécuter la commande vous-même). Cependant, elle peut être très utile, car le nom de fichier indique l'ID de session d'où il a été extrait afin de vous permettre de trouver le malware. Vous pouvez alors interroger cet ID de session et rechercher l'hôte éventuellement infecté par ce malware afin d'entreprendre une action. Les paramètres `includeFileTypes` ou `excludeFileTypes` vous permettent de filtrer ce qui doit être extrait (voir l'aide associée à la commande). Par exemple, l'ajout de `excludeFileTypes=".exe;.dmg;.msi"` empêche l'extraction des exécutables et des programmes d'installation. Cette commande se contente d'extraire des fichiers en nonstop de toutes les sessions existantes et nouvelles. Lorsque le répertoire contient plus de 200 Mio de fichiers, elle nettoie automatiquement les fichiers toutes les 10 minutes.

Note: Cette commande est seulement valable pour les sessions de paquets et non les logs.

```
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
```

Cette commande extrait les fichiers des sessions HTTP et de messagerie sur une période d'une heure et groupe ensuite les fichiers extraits en répertoires spécifiés par le paramètre `subdirFileTypes`. Par exemple, tout fichier audio extrait qui a l'extension .wav, .mp3 ou .aac est placé dans le sous-répertoire audio qui est créé dans le répertoire de sortie spécifié. Il en va de même pour tous les autres groupes spécifiés dans ce paramètre. Selon leur extension, certains fichiers sont aussi renommés automatiquement. Cette opération est effectuée par `renameFileTypes`. Tous les fichiers avec l'extension .download, .octet-stream ou .program sont renommés .exe. Les fichiers avec l'extension .jpeg sont renommés jpg. Lorsque le répertoire de premier niveau dépasse 500 Mio, les fichiers les plus anciens sont effacés. Cette commande s'arrête à la dernière session qui porte l'heure à laquelle la commande a démarré.

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

Cette commande recherche tous les paquets et logs (le paramètre `sp`) correspondant au mot-clé `party`. Si le mot-clé party est trouvé quelque part dans les paquets et les logs, elle génère l'ID de session accompagné du texte qu'elle a trouvé et du texte autour comme contexte. La clause where indique qu'elle explore uniquement le trafic Web et de

messagerie. Le paramètre `ci` indique qu'il s'agit d'une recherche dans laquelle les minuscules ne sont pas différenciées des majuscules. Vous pouvez remplacer `keyword` par `regex` pour obtenir une recherche regex.

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

C'est un exemple de commande intéressant. Cette commande explore tous les logs (ou éventuellement les paquets) pour rechercher le mot-clé `checkpoint`, et, lorsque ce mot-clé est trouvé, elle extrait le log dans un fichier **checkpoint-logs.log**. Cette commande s'accompagne de toutes sortes de possibilités. À la base, lorsqu'une occurrence est détectée, la commande passe la session à l'appel de contenu. Ainsi, les paramètres que vous transmettez à la `sdk search` qui ne sont pas reconnus sont passés à l'appel de contenu. Cela permet d'exploiter toutes les capacités de l'appel de `sdk content`, mais seulement en travaillant sur des sessions contenant des occurrences de contenu recherché. Avec le pouvoir vient la responsabilité !

RSA

Commandes utilisées pour le dépannage

NwConsole propose les commandes suivantes, utiles au dépannage Security Analytics :

- **whatIsWrong** : fournit un snapshot de la configuration du service, des statistiques et des logs liés aux échecs et avertissements, sur une période de temps spécifiée.
- **dbcheck** : vérifie la cohérence des fichiers de la base de données.
- **topQuery** : aide à identifier des requêtes qui prennent très longtemps à exécuter.
- **netbytes** : dépanne les connexions réseau sur l'hôte actuel.
- **netspeed** : dépanne la connexion entre l'ordinateur hôte exécutant NwConsole et l'ordinateur cible qui y est connecté grâce à la commande `login`.

Les sections suivantes, ainsi que les pages d'informations (man) et d'aide NwConsole, donnent des informations supplémentaires.

whatIsWrong

Lorsqu'un service ne fonctionne pas correctement, la raison se trouve généralement dans les logs écrits par le service. Vous pouvez utiliser la commande de la console `whatIsWrong` pour obtenir un snapshot de la configuration du service, des statistiques et des logs liés aux échecs et avertissements (avec les logs du contexte environnant) pour la période de temps spécifiée. Par défaut, il s'agit des sept derniers jours. Vous pouvez enregistrer les résultats de la commande whatIsWrong dans le fichier de texte brut spécifié. La sortie de cette commande peut être un point de départ utile pour vous aider à déterminer le problème du service.

Pour utiliser la commande whatIsWrong de la console, connectez-vous au service à dépanner via la commande `login` et exécutez la commande `whatIsWrong`.

Conseil : Utilisez `help whatIsWrong` pour voir tous les paramètres disponibles, notamment la période de jours/heures sur laquelle consulter les événements, le chemin d'accès pour stocker les résultats, si vous souhaitez ou non ajouter ou remplacer le fichier de résultats, et le délimiteur à utiliser pour les champs du fichier log. Vous pouvez également limiter les résultats aux logs les plus récents permettant de trouver du contexte, et spécifier le nombre de logs de contexte à récupérer par avertissement/échec.

Chaque fois que vous recevez une requête de logs pour un service Core, vous devez d'abord exécuter la commande whatIsWrong et utiliser les résultats collectés en tant que point de départ.

Last Modified: August 10 2016, 9:33AM

dbcheck

La commande `dbcheck` est utilisée pour vérifier la cohérence des fichiers de la base de données (session, méta, paquets, logs, statistiques, etc). Cela peut être nécessaire lorsqu'un service ne peut pas se lancer en raison d'erreurs de cohérence des fichiers de la base de données. Généralement, le service est récupéré automatiquement et corrige tout problème de cohérence au démarrage, mais il arrive que cela ne se produise pas. Lorsqu'un service démarre (comme un Decoder), il ne lit ou n'ouvre pas la plupart des fichiers de la base de données afin de se lancer rapidement. Il considère que la plupart des fichiers sont cohérents et n'effectue qu'une vérification rapide des fichiers écrits en dernier. En cas de problème, `dbcheck` peut réaliser ces vérifications de cohérence, mais SEULEMENT si le service n'est pas en cours d'exécution.

⚠ **Caution:** N'essayez pas d'exécuter cette commande alors qu'un service est en cours de fonctionnement.

Par exemple, vous pouvez vérifier un fichier unique :

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

Vous pouvez également utiliser des caractères génériques pour vérifier plusieurs fichiers :

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

topQuery

La commande `topQuery` peut aider à identifier les requêtes qui prennent trop de temps à s'exécuter. Cette commande analyse les fichiers Audit Log d'un service et renvoie les N requêtes en cours d'exécution qui prennent le plus de temps pour la période spécifiée.

La méthode la plus facile pour l'exécuter est de se connecter au service (généralement un Broker ou un Concentrator) et de saisir `topQuery`. Le comportement par défaut est de renvoyer les 100 requêtes qui ont pris le plus de temps à s'exécuter au cours des sept derniers jours.

Saisissez `help topQuery` pour la liste de paramètres. Voici quelques exemples supplémentaires avec des explications :

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

Cette commande renvoie les 10 principales requêtes au cours des 12 dernières heures.

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

Cette commande renvoie les 100 principales requêtes entre le 1er mars 2015 et le 14 mars 2015. Les heures sont en heures UTC, non locales.

Last Modified: August 10 2016, 9:33AM

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

Au lieu de se connecter à un service, elle analyse les messages d'audit syslog pour les 20 principales requêtes au cours des 7 derniers jours, mais uniquement pour les requêtes exécutées par l'utilisateur sauser1. Elle écrit les 20 principales requêtes dans le fichier /tmp/top20.txt plutôt que sur l'écran de la console. L'utilisateur du paramètre est un regex, donc vous pouvez spécifier plusieurs noms d'utilisateur en écrivant par exemple `user="(sauser1|sauser2)"`.

netbytes

La commande `netbytes` est très utile pour dépanner les connexions réseau sur l'hôte actuel. Elle affiche des statistiques continues d'envoi et de réception pour toutes les interfaces réseau. Une fois exécutées, vous devez appuyer sur **Ctrl-C** pour quitter cette commande, qui permet également de quitter NwConsole.

netspeed

La commande `netspeed` est utilisée pour dépanner la connexion entre l'ordinateur hôte qui exécute NwConsole et l'ordinateur distant qui y est connecté via la commande `login`. Vous devez préciser la quantité d'octets à transférer, ce qui définira la vitesse de la connexion. La commande netspeed est très utile pour le dépannage des problèmes de performances d'agrégation qui peuvent être liés au réseau.

```
login somedecoder:50004 admin ...
```

```
netspeed transfer=4g
```

Pour dépanner la connexion entre un Concentrator et un Decoder, ouvrez une session SSH dans le Concentrator, exécutez NwConsole, puis connectez-vous au Decoder et exécutez `netspeed`. La sortie de la commande vous donne une indication du débit réseau maximum. Si la valeur est bien inférieure à l'interface standard d'1 Gbit/s, il peut s'agir d'un problème de réseau.

Last Modified: August 10 2016, 9:33AM