

RSA

NETWITNESS®
PLATFORM

Meta Guide



Copyright © 1994-2022 Dell Inc. or its subsidiaries. All Rights Reserved.

Trademarks

RSA, the RSA Logo and EMC are either registered trademarks or trademarks of EMC Corporation in the United States and/or other countries. All other trademarks used herein are the property of their respective owners. For a list of EMC trademarks, go to www.emc.com/legal/emc-corporation-trademarks.htm.

License Agreement

This software and the associated documentation are proprietary and confidential to EMC, are furnished under license, and may be used and copied only in accordance with the terms of such license and with the inclusion of the copyright notice below. This software and the documentation, and any copies thereof, may not be provided or otherwise made available to any other person.

No title to or ownership of the software or documentation or any intellectual property rights thereto is hereby transferred. Any unauthorized use or reproduction of this software and the documentation may be subject to civil and/or criminal liability. This software is subject to change without notice and should not be construed as a commitment by EMC.

Third-Party Licenses

This product may include software developed by parties other than RSA.

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.

January 2022

Contents

Custom CEF Parser	7
Context	7
Functionality and Characteristics	7
Details	7
Create Custom CEF Parser	8
Add Vendor, Product, Device, and Group Definition	8
Override Existing Device Definitions	8
Override Existing CEF Tag to NetWitness Meta Tag Mapping	8
How metaName Works	9
Override Existing CEF Tag to NetWitness Meta Tag Mapping For a Specific Device	9
Deploy the Custom CEF Parser	10
Live Content Search Tags	11
Context	11
Context Hub Lists in ESA Rules	13
Use CH Lists in ESA Rules	13
OOTB Context Hub Lists	14
How to Update a Context Hub List	16
How to Create a Context Hub List	18
How to Add a Context Hub List as an Enrichment source	19
Create an ESA Rule that Uses a Context Hub list	21
Example of an ESA Rule that Uses a CH list	22
EPL Syntax for whitelists and Blacklists	24
Known Limitations	25
Can the Context Hub lists comparison be case-insensitive?	25
What are the limitations between Basic Rule Builder and Live / Advanced Rules?	25
What happens when you deploy an 11.1 CH List ESA rule to version prior to 11.1?	25
HTTP Lua Parser Options File	26
registerURL	27
splitQuery	27
useOrigIP	27

refererPath	28
userAgent	28
respReason	28
decompress	29
advanced	30
customHeaders	30
NetWitness Investigation Model	32
Model Hierarchy	33
Threat	33
Examples	33
Attack Phase	34
Malware	35
Identity	35
Examples	36
Authentication	36
Authorization	36
Accounting	37
Behavior Analytics	37
Assurance	38
Examples	38
Governance	39
Risk	39
Compliance	40
Operations	40
Examples	41
Situation Awareness	41
Event Analysis	41
LDAP Parser Options File	43
ports	43
idOnly	44
parseResponses	44
Log Parser Customization	45
Loading Order	45
File Location and Naming	45
Header and Message Duplication	45

Examples	46
Example Code	46
Common Steps	46
Add a New Item	47
Add New Header	47
Add New Message	47
Add New Valuemap	48
Add New Tagval	49
insertBefore and insertAfter	50
Modify an Existing Item	50
Modify Header	50
Modify Message	50
Modify Valuemap	51
Modify Tagval	51
Lua Debugging Tool	53
Mail Lua Parser Options File	54
registerEmailSrcDst	54
parseQuoted	55
registerAddressHosts	55
parseReceived	55
Packet Parsers	56
Context	56
Packet Parsers in NetWitness Suite	56
Discontinued Packet Parsers	64
Phishing Lua Parser Options File	66
Deduplicate Host Registration	67
Check Host Consistency	67
Whitelist Domain	67
Register URL Components	67
Register Entire URL	68
Host Key	68
SMTP Lua Parser Options File	69
registerEmailSrcDst	69
registerAddressHosts	70

errorCodeOnly	70
System Parsers	71
Context	71
System Parsers in RSA NetWitness Platform	71
RSA Threat Content mapping with MITRE ATT&CK™	74
Introduction to MITRE ATT&CK™ Navigator	74
Generate of MITRE ATT&CK™ Metadata for RSA NetWitness Content	74
Configure RSA NetWitness for Mitre ATT&CK™ Metadata	76
TLD Lua Parser Options File	80
deduplicate	80
localDomains	81
Traffic Flow Lua Parser	82
Introduction	82
Setup	83
Deploy to Network Decoders	83
Reload Parsers:	85
Deploy to Log Decoders (For versions Prior to 11.0)	86
Update XML Files	88
Add Entries to the Index Concentrator File	88
Add Entries to the Table Map File (Log Decoders Only)	88
Update ESA Configuration	89
Tuning	89
Parser Defaults	89
Options File	90
Editing the Options File	91
Option File Details	91
Matching Rules	92
Examples	93
Key Mappings and Defaults	95
Configure Windows Collection	97
Overview	97
Create a User Account for RSA NetWitness Platform	100

Custom CEF Parser

This topic discusses and describes the custom CEF parser (**cef-custom.xml**), that overrides the standard, base CEF (Common Event Format) parser.

Context

Customers need the ability to customize the key mappings in the base CEF parser, based on their requirements. If they customize the base CEF parser, their changes will be lost when updating to the latest content.

The Custom CEF parser maintains the customizations separate from the base parser information.

Functionality and Characteristics

The Custom CEF parser has the following capabilities:

- You can override the base mapping for a tag to a different meta key.
- You cannot override the base Header and Message Definitions
- The Custom CEF parser is a super set of the base CEF parser. This means:
 - Any new tags you define are appended to the set of existing tags in the base parser.
 - If there is a conflict (that is, if the custom parser has a different mapping for a given tag than the base parser), the custom parser "wins." For all such conflicts, the mapping defined for the custom parser overrides the mapping in the base parser.
- If the custom parser definition is invalid and throws an error, the base parser is used, and the error is reported.

Details

RSA NetWitness Platform Log Decoders version 10.6.4 and newer support both the custom and base CEF parsers. Upon starting, a Log Decoder service will read both the base CEF parser and the custom CEF parser. It will override base parser behavior as described above. The Log Decoder service maintains the mappings in the custom CEF parser even when a newer version of the CEF parser is downloaded from Live.

The Custom CEF parser supports overriding and adding new entries for the following keys:

- VendorProducts key
- ExtentionKey

- device2meta key
- CN and CS keys

Create Custom CEF Parser

This section walks through examples for how to create a custom CEF parser.

Note: A sample CEF custom file is available on RSA Link here:

<https://community.rsa.com/docs/DOC-79227>

Add Vendor, Product, Device, and Group Definition

To map a log of a product called **Product1** from a vendor called **Vendor1** to device name **Device1** of **Group1**, create a new **Vender2Device** tag in the `cef-custom.xml` file as shown here:

```
<DEVICEMESSAGES>
  <VendorProducts>
    <Vendor2Device vendor="Vendor1" product="Product1" device="Device1"
group="Group1"/>
  </VendorProducts>
</DEVICEMESSAGES>
```

Override Existing Device Definitions

To change an existing **Vender2Device** definition, create an overriding **Vender2Device** tag in the `cef-custom.xml` file as shown below.

The following code is in `cef.xml`:

```
<VendorProducts>
  <Vendor2Device vendor="RSA" product="Security Analytics NetFlow Collector"
device="rsaflow" group="Switch"/>
</VendorProducts>
```

To change the device name from **rsaflow** to **My Own Device**, add the following code to `cef-custom.xml`:

```
<DEVICEMESSAGES>
  <VendorProducts>
    <Vendor2Device vendor="RSA" product="Security Analytics NetFlow Collector"
device="My Own Device" group="switch"/>
  </VendorProducts>
</DEVICEMESSAGES>
```

Override Existing CEF Tag to NetWitness Meta Tag Mapping

To change existing CEF tag to NetWitness Meta key mapping defined in **ExtentionKey**, create an overriding **ExtentionKey** tag in the `cef-custom.xml` file as shown below.

The following code is in `cef.xml`:


```
:  
  <ExtensionKey cefName="dst" metaName="daddr"/>  
:
```

To change the CEF tag **dst** to be mapped to a new key, **forward.ip**, instead of the original **daddr** add the following code to `cef-custom.xml`:

```
<DEVICEMESSAGES>  
  <ExtensionKeys>  
    <ExtensionKey cefName="dst" metaName="forward.ip"/>  
  </ExtensionKeys>  
</DEVICEMESSAGES>
```

Note: This change affect all devices.

How metaName Works

The `metaName` holds the name of the key that represents the *log parser key name*. The table map file maps the log parser key to the *meta key*. The log parser key is used in the parser, and the meta key is available in Investigator.

The [Unified Data Model](#) describes the NetWitness Suite data model, as well as how meta flows through RSA NetWitness Platform.

Override Existing CEF Tag to NetWitness Meta Tag Mapping For a Specific Device

To change existing CEF tag to NetWitness Meta key mapping defined in `ExtentionKey` for just one device, create a new or an overriding **device2meta** tag in `cef-custom.xml` as shown below.

The following code is in `cef.xml`:

```
:  
<ExtensionKey cefName="proto" metaName="protocol">  
  <device2meta device="rsaflow" metaName="ip_proto"/>  
</ExtensionKey>  
:
```

To change the CEF tag **proto** to be mapped to a new key, **proto1**, instead of the original **ip_proto** for the **rsaflow** device, add the following code to `cef-custom.xml`:

```
<DEVICEMESSAGES>  
  <ExtensionKeys>  
    <ExtensionKey cefName="proto" metaName="protocol">  
      <device2meta device="rsaflow" metaName="proto1"/>  
    </ExtensionKey>  
  </ExtensionKeys>  
</DEVICEMESSAGES>
```

In this case, you are changing CEF tag **proto** to be mapped to a new key, **proto1**, instead of the original, **ip_proto**, for device "rsaflow".

Note: Use of this new meta key requires adjustment to the `table-map-custom.xml`. This change does not affect any devices other than `rsaflow`.

Deploy the Custom CEF Parser

Follow these steps to deploy the custom CEF parser.

1. Create and add mappings to the `cef-custom.xml` file.
2. Upload the custom CEF parser to your Log Decoder services. Upload the file to the following directory, overwriting the existing file:

```
/etc/netwitness/ng/envision/etc/devices/cef
```

Note: Use an SSH tool, for example WindSCP, to copy your custom file to the Log Decoder folder.

3. If necessary, update the `table-map-custom.xml` file.
4. Reload the CEF parser.

Live Content Search Tags

This topic describes the Advanced Security Operations Center (ASOC) tags. These tags are used to organize Live content and to deliver an accurate path to information security incident response. The tags are found in the Live Search view, as:

- **Tags** in Security Analytics 10.x
- **Categories** in NetWitness Suite 11.x

Context

The objective of a tag is to catalog existing content for deployment according to an incident response approach. Currently, the model contains the following tags:

- accounting
- action on objectives
- application analysis
- assurance
- attack phase
- audit
- authentication
- authorization
- command and control
- compliance
- corporate
- crimeware
- data exfiltration
- data sabotage
- delivery
- denial of service
- event analysis
- exploit

- featured
- file analysis
- filters
- flow analysis
- identity
- installation
- key loggers
- lateral movement
- log analysis
- malware
- malware analysis
- operations
- organizational hazard
- protocol analysis
- reconnaissance
- remote access trojans
- risk
- situation awareness
- spectrum
- threat
- vulnerability management
- web shells

These tags are a part of the investigation model described in the [NetWitness Investigation Model](#).

Note: When you search in Live, note that categories or tags you enter are ORed. That is, if you search for **threat** and **assurance**, all content that is tagged as either **threat** or **assurance** is returned.

Example: [Live Search in NetWitness Suite 11.x](#) 11.x, or [Live Search in Security Analytics 10.x](#).

Context Hub Lists in ESA Rules

For RSA NetWitness Platform 11.1 and later, ESA Rules can use Context Hub (CH) Lists as whitelists and blacklists in their construction and processing. To see details about these rules, see [RSA ESA Rules](#).

This topic discusses the following:

- [Use CH Lists in ESA Rules](#)
- [OOTB Context Hub Lists](#)
- [How to Update a Context Hub List](#)
- [How to Create a Context Hub List](#)
- [How to Add a Context Hub List as an Enrichment source](#)
- [Create an ESA Rule that Uses a Context Hub list](#)
- [Example of an ESA Rule that Uses a CH list](#)
- [EPL Syntax for whitelists and Blacklists](#)
- [Known Limitations](#)

Use CH Lists in ESA Rules

As of RSA NetWitness 11.1, Context Hub lists can be used in the processing of ESA Rules.

1. Configure an existing CH list, or create and configure your own CH list. Basically, you need to add a list of values to either an existing CH list or create your own and then add values.
2. Configure the CH List within ESA by adding it as an Enrichment source.
3. Load the CH list into an ESA Rule when you build statements and define the rule.

An advantage of using CH lists in ESA rules, is that from the Respond and Investigate screens in NetWitness Suite, you can right-click on an item and update the list on-the-fly. For the selected item, you can add it to or remove it from any of your CH lists.

For details, see the following documentation in the [RSA NetWitness Logs & Network 11.x Documentation](#) space on RSA Link:

- Investigate: "[Manage Context Hub Lists and List Values](#)" topic in Investigate topic in the *NetWitness Investigate User Guide*
- Respond: "[Context Lookup Panel](#)" or "[Investigate the Incident](#)" topics in the *NetWitness Respond User Guide*

OOTB Context Hub Lists

The following Context Hub lists are available out of the box in RSA NetWitness 11.1. They are delivered empty: users need to configure the lists by adding entries.

Without this configuration step, the rules may not deliver results. You can add entries to the lists manually, or through import of CSV files. For details, see [Configure Lists as a Data source](#) in the *Context Hub Guide*.

The following lists are delivered with RSA NetWitness 11.1:

- **User_Whitelist:** A list of users that should be excluded from monitoring within rules configured to use it.
- **User_Blacklist:** A list of users that should be included for monitoring within rules configured to use it.
- **Admin_Accounts:** A list of privileged user accounts that should be included for monitoring within rules configured to use it.
- **Service_Accounts:** A list of service accounts that should be included for monitoring within rules configured to use it.
- **Guest_Accounts :** A list of guest user accounts that should be included for monitoring within rules configured to use it.
- **Domain_Controllers:** A list of domain controllers that should be included for monitoring within rules configured to use it.
- **Host_Whitelist:** A list of host names that should be excluded from monitoring within rules configured to use it.
- **Host_Blacklist:** A list of host names that should be included for monitoring within rules configured to use it.
- **IP_Whitelist:** A list of IP addresses that should be excluded from monitoring within rules configured to use it. CIDR notation and regular expressions may not be used.
- **IP_Blacklist:** A list of IP addresses that should be included for monitoring within rules configured to use it. CIDR notation and regular expressions may not be used.

The following table lists the rules that use each of the CH Lists.


CH List Name	ESA Rules that Use the List
User_Whitelist	<p>Logins Across Multiple Servers</p> <p>User Login Baseline</p> <p>Failed logins Followed By Successful Login and a Password Change</p> <p>Windows Suspicious Admin Activity: Firewall Service Stopped</p> <p>Windows Suspicious Admin Activity: Shared Object Accessed</p> <p>User Added to Admin Group Same User Login OR Same User su sudo</p> <p>Multiple Successful Logins from Multiple Diff Src to Diff Dest</p> <p>Multiple Successful Logins from Multiple Diff Src to Same Dest</p> <p>Multiple Failed Logins from Same User Originating from Different Countries</p> <p>Insider Threat Mass Audit Clearing</p> <p>Multiple Account Lockouts from Same or Different Users</p> <p>Multiple Failed Logins Followed by Successful Login</p> <p>Windows Suspicious Admin Activity: Audit log Cleared</p> <p>Windows Suspicious Admin Activity: Network Share Created</p> <p>User Account Created and Deleted Within an Hour</p> <p>Multiple Failed Logins from Multiple Diff Sources to Same Dest</p> <p>User added to administrative group then SIGHUP detected</p> <p>Multiple Failed Logins from Multiple Users to Same Destination</p> <p>Failed logins Outside Business Hours</p>
User_Blacklist	<p>Direct Login By A Watchlist Account</p>
Admin_Accounts	<p>Privilege User Account Password Change</p> <p>Privilege Escalation Detected</p> <p>Suspicious Privileged User Access Activity</p> <p>Multiple Failed Privilege Escalations by the Same User</p> <p>Multiple Login Failures by Administrators to Domain Controller</p>
Guest_Accounts	<p>Multiple Login Failures by Guest to Domain Controller</p>
Host_Whitelist	<p>Multiple Failed Logins from Multiple Diff Sources to Same Dest</p> <p>Multiple Successful Logins from Multiple Diff Src to Diff Dest</p> <p>Multiple Successful Logins from Multiple Diff Src to Same Dest</p> <p>Multiple Failed Logins from Multiple Users to Same Destination</p> <p>Lateral Movement Suspected Windows</p>

CH List Name	ESA Rules that Use the List
Host_Blacklist	krbtgt Account Modified on Domain controller Multiple Login Failures by Administrators to Domain Controller Multiple Login Failures by Guest to Domain Controller
IP_Whitelist	Multiple Failed Logins from Multiple Diff Sources to Same Dest Multiple Successful Logins from Multiple Diff Src to Diff Dest Multiple Successful Logins from Multiple Diff Src to Same Dest Multiple Failed Logins from Multiple Users to Same Destination
IP_Blacklist	krbtgt Account Modified on Domain controller Multiple Login Failures by Administrators to Domain Controller Multiple Login Failures by Guest to Domain Controller

How to Update a Context Hub List

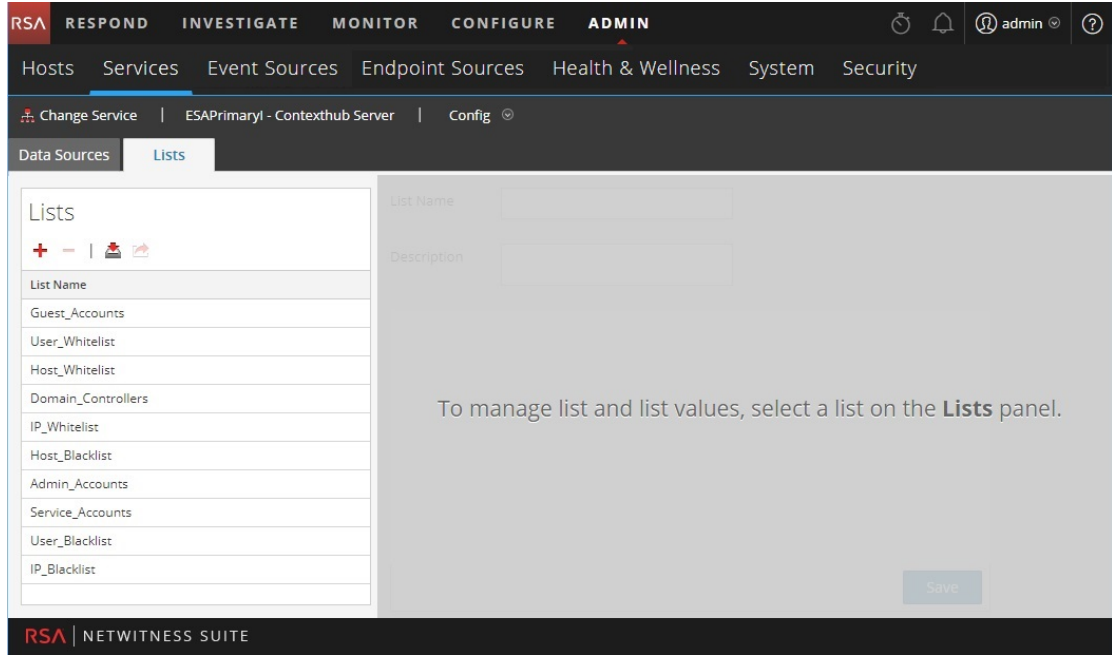
1. Go to **ADMIN > Services**.

The services view is displayed.

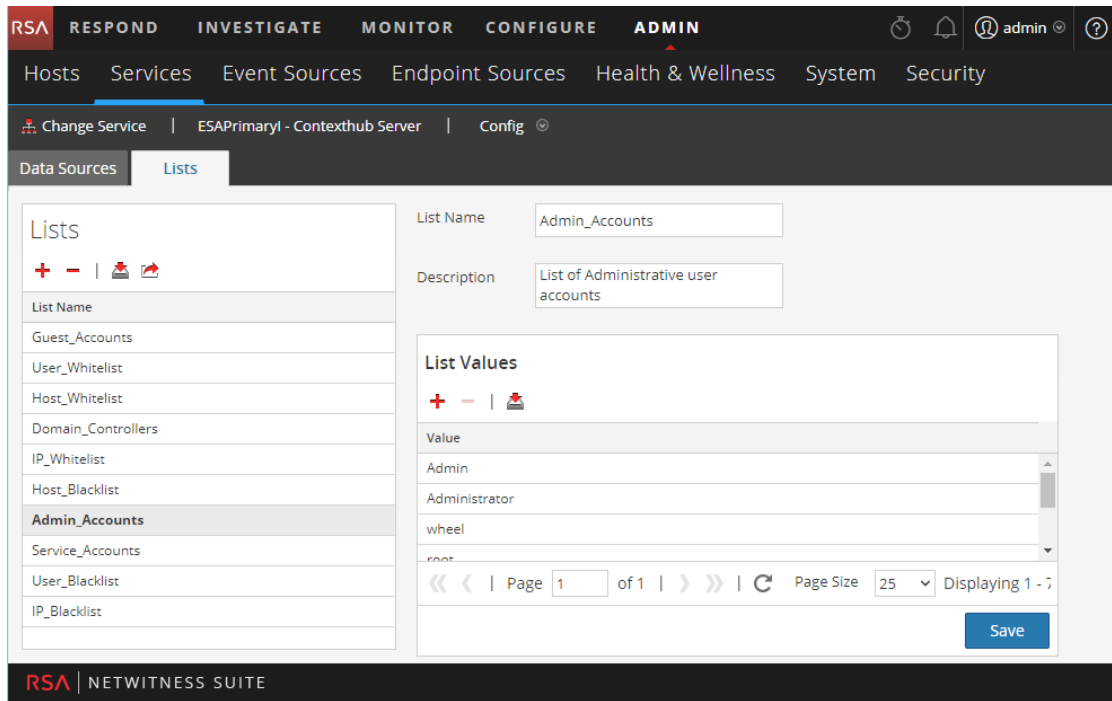
2. Select the Context Hub service and click  > **View > Config**.

The Services Config View of Context Hub is displayed.




3. Select the Lists tab.



4. In the **Lists** tab, select the list that you wish to update.





5. In the List Values section, there are controls for adding and removing items, as well as for importing a list.

- To add an entry: click  then enter a new value.
 - To remove an entry: select it then click .
 - To import a list, click , then navigate to a CSV file that contains the entries for your list.
6. Do either of the following:
- Click **Save** to save your changes, or
 - Click anywhere outside the **List Values** section to discard your changes. You receive a confirmation message asking you to make sure you want to discard your changes: click **Yes** to discard your changes or **No** to go back to the screen with your unsaved changes.



For more information, see the topic "Configure Lists as a Data source" in the *Context Hub Configuration Guide* in [RSA NetWitness Platform space](#) on RSA Link.

How to Create a Context Hub List

Creating a list is very similar to updating an existing list.

1. Go to **ADMIN > Services**.
2. Select the Context Hub service and click  > **View > Config**.
3. Select the **Lists** tab.
4. In the **Lists** tab, click , then enter a name for your list.

Note: Make sure the name does not contain spaces. If the name of a list contains spaces, it cannot be used in an ESA Rule.

5. Add values to the list, or import an existing list:
 - To add an entry: click  then enter a new value.
 - To import a list, click , then navigate to a CSV file that contains the entries for your list.
6. Click **Save** to save your new list.

How to Add a Context Hub List as an Enrichment source

If you add a new CH list, before you can use it in an ESA Rule, you need to add it as an enrichment source.

1. Go to **CONFIGURE > ESA Rules**.
2. Select the **Settings** tab, then **Enrichment sources**.

Enabled	Name ^	Type	Description	Last Modified	Actions
<input checked="" type="checkbox"/>	Admin_Accounts	Context Hub		2018-02-06 19:39:55	
<input checked="" type="checkbox"/>	Default_GeoIP	GeoIP	Default Geo IP Enrichme...	2018-02-02 15:28:47	
<input checked="" type="checkbox"/>	Domain_Controllers	Context Hub		2018-02-09 16:17:24	
<input checked="" type="checkbox"/>	Guest_Accounts	Context Hub		2018-02-07 17:30:07	
<input checked="" type="checkbox"/>	User_Blacklist	Context Hub		2018-02-02 18:23:15	
<input checked="" type="checkbox"/>	User_Whitelist	Context Hub		2018-02-12 19:48:51	

- Click > **Context Hub**.

The Context Hub List dialog box is displayed.

- Select a list, add a description, and select a column.

Context Hub List

Enable

Select List

Description

Columns

Name

LIST

Page To Local Store




[For information on how to define a Context Hub List, see the documentation](#)

Cancel Save

- Click **Save** to finish.

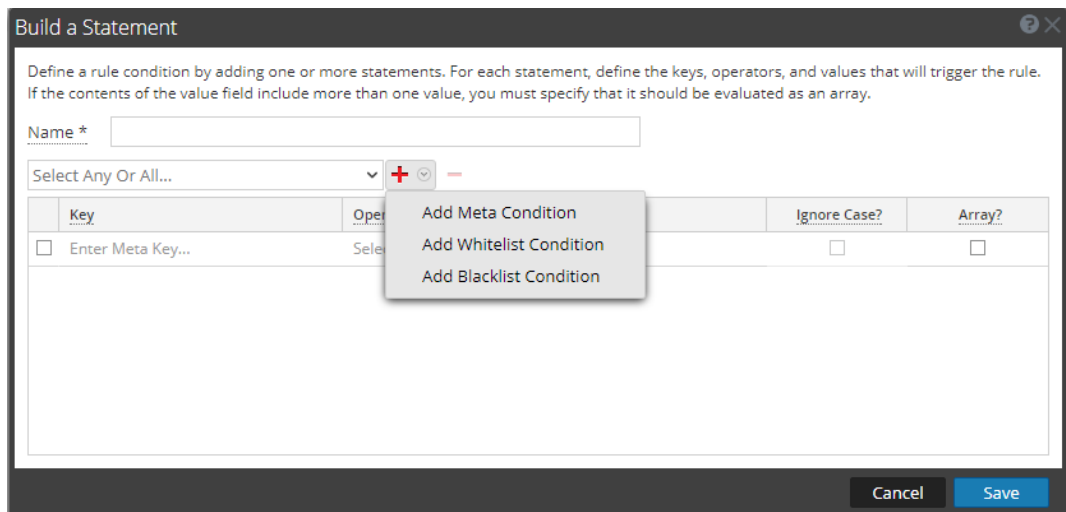
For more information, see the topic "Configure Context Hub List as an Enrichment source " in the *Alerting with ESA Correlation Rules User Guide* in [RSA NetWitness Platform space](#) on RSA Link.

Create an ESA Rule that Uses a Context Hub list

1. Go to **CONFIGURE > ESA Rules**.
2. In the **Rules** tab, click   > **Rule Builder**.
A New Rule tab opens.
3. In the New Rule tab, enter a name and description.
4. In the **Conditions** section, click  to open the **Build a Statement** dialog box.
5. You can add a whitelist, blacklist, or meta condition. This procedure details adding a list, so choose either:
 - Add whitelist Condition, or
 - Add Blacklist Condition

In this example, we add a whitelist condition.

- a. Click   > **Add whitelist Condition**.



- b. In the Key column, from the drop-down menu, select a whitelist to use, for example **User_Whitelist**.

Build a Statement

Define a rule condition by adding one or more statements. For each statement, define the keys, operators, and values that will trigger the rule. If the contents of the value field include more than one value, you must specify that it should be evaluated as an array.

Name *

Select Any Or All...

<input type="checkbox"/>	Key	Operator	Value	Ignore Case?	Array?
<input type="checkbox"/>	Enter Meta Key...	Select Operator..	Enter Value	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	whitelist.User_Whitelist				
<input type="checkbox"/>	Enter Field Name...	is	Select...		

Whitelist conditions can be added to exclude only those items defined in an enrichment list. Map a list column to an event meta key to join the list to the incoming data stream.

- c. Select a column name from the list, then select an operator and enter the meta value for the corresponding value field.

Build a Statement

Define a rule condition by adding one or more statements. For each statement, define the keys, operators, and values that will trigger the rule. If the contents of the value field include more than one value, you must specify that it should be evaluated as an array.

Name *

Select Any Or All...

<input type="checkbox"/>	Key	Operator	Value	Ignore Case?	Array?
<input type="checkbox"/>	Enter Meta Key...	Select Operator..	Enter Value	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	whitelist.User_Whitelist				
<input checked="" type="checkbox"/>	LIST	is	event.user_dst		

Whitelist conditions can be added to exclude only those items defined in an enrichment list. Map a list column to an event meta key to join the list to the incoming data stream.


- d. Click **Save** to save the statement and close the dialog box.

6. Continue defining the rule until it is complete. For details, see "Add a Rule Builder Rule" in the *Alerting Using ESA Guide*.

Example of an ESA Rule that Uses a CH list

The **Failed Logins Followed By Successful Login Password Change** ESA rule uses the User_Whitelist context hub list.

You can view the syntax in RSA NetWitness Suite:

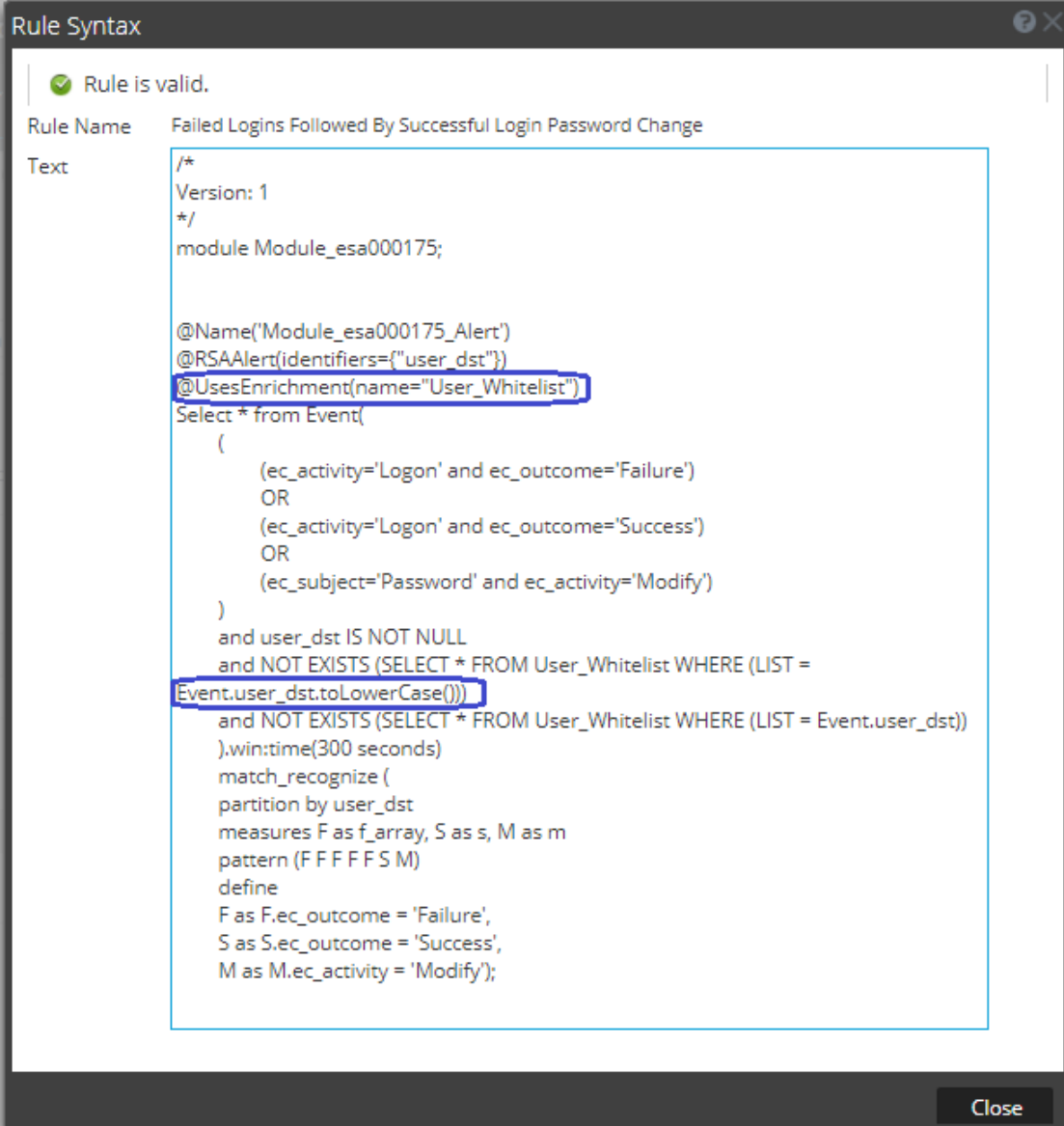
1. Go to **CONFIGURE > ESA Rules**.
2. In the **Rules** tab, select the **Failed Logins Followed By Successful Login Password Change** rule and click .

A tab for editing the rule is displayed.

3. Scroll down to the bottom of the page and click **Show Syntax**.

The Rule Syntax dialog box is displayed.

4. Look over the syntax to get a sense of the EPL for this rule. When finished, click **Close** to close the Rule Syntax dialog box.



The screenshot shows a dialog box titled "Rule Syntax" with a green checkmark and the text "Rule is valid." The dialog displays the following information:

Rule Name: Failed Logins Followed By Successful Login Password Change

Text:

```
/*
Version: 1
*/
module Module_esa000175;

@Name('Module_esa000175_Alert')
@RSAAlert(identifiers={"user_dst"})
@UsesEnrichment(name="User_Whitelist")
Select * from Event(
  (
    (ec_activity='Logon' and ec_outcome='Failure')
    OR
    (ec_activity='Logon' and ec_outcome='Success')
    OR
    (ec_subject='Password' and ec_activity='Modify')
  )
  and user_dst IS NOT NULL
  and NOT EXISTS (SELECT * FROM User_Whitelist WHERE (LIST =
Event.user_dst.toLowerCase()))
  and NOT EXISTS (SELECT * FROM User_Whitelist WHERE (LIST = Event.user_dst))
).win:time(300 seconds)
match_recognize (
  partition by user_dst
  measures F as f_array, S as s, M as m
  pattern (F F F F S M)
  define
  F as F.ec_outcome = 'Failure',
  S as S.ec_outcome = 'Success',
  M as M.ec_activity = 'Modify');
```

A "Close" button is located at the bottom right of the dialog.

EPL Syntax for whitelists and Blacklists

A whitelist ("known good") is a list of event meta value to exempt from alerts.

Whitelist Example Syntax (in bold):

```
@RSAAlert(oneInSeconds=0, identifiers={"user_dst"})
@UsesEnrichment(name="User_Whitelist")
SELECT * FROM
    Event (
        medium = 32
        AND ec_activity = 'Logon'
        AND ec_outcome = 'Success'
        AND logon_type IN ('2','10','11','12')
        AND device_class = 'Windows Hosts'
        AND reference_id IN ('4624', '528', '540')
        AND user_dst IS NOT NULL
        AND NOT EXISTS (SELECT * FROM User_Whitelist WHERE (LIST =
            Event.user_dst.toLowerCase()))
        AND NOT EXISTS (SELECT * FROM User_Whitelist WHERE (LIST =
            Event.user_dst))
    );
```

A Blacklist ("known bad") is a list of event meta value used to trigger alerts.

Blacklist Example Syntax (in bold):

```
@RSAAlert(oneInSeconds=0, identifiers={"user_dst"})
@UsesEnrichment(name="User_Blacklist")
SELECT * FROM
    Event (
        medium = 32
        AND ec_activity = 'Logon'
        AND ec_outcome = 'Success'
        AND logon_type IN ('2','10','11','12')
        AND device_class = 'Windows Hosts'
        AND reference_id IN ('4624', '528', '540')
        AND user_dst IS NOT NULL
        AND
        (
            EXISTS (SELECT * FROM User_Blacklist WHERE (LIST
                = Event.user_dst.toLowerCase()))
            OR
            EXISTS (SELECT * FROM User_Blacklist WHERE (LIST
                = Event.user_dst))
        )
    );
```


If you create your own rules using CH lists, make sure to the **UsesEnrichment()** statement, as shown in the above example:

```
@UsesEnrichment(name="User_Whitelist")
```

In this example, we are loading the **User_Whitelist** into the system for this rule.

Note: It is fine to have the same list loaded (that is, named in multiple UsesEnrichment statements) in multiple deployed ESA Rules. The system only loads each CH list once.

Use the **toLowerCase()** function to convert the received meta to all lower case.

```
Event.user_dst.toLowerCase()
```

In the above example, the **user_dst** meta values are converted to all lowercase. If you have created your CH lists so that all entries are also in all lowercase, your comparison is case-insensitive.

Known Limitations

Can the Context Hub lists comparison be case-insensitive?

In order to get case-insensitive matching between CH lists and event meta, customers must add users within the CH lists as all lower case. Context hub lists do not have the ability to make the entries lower case before performing the match. Additionally, be sure to use the **toLowerCase()** function in your rules, so that the meta values are converted to all lowercase for the comparison.

What are the limitations between Basic Rule Builder and Live / Advanced Rules?

Only able to use a single whitelist or blacklist within the basic rule builder.

What happens when you deploy an 11.1 CH List ESA rule to version prior to 11.1?

The rule will be unable to deploy, it will be disabled, and an error will be written to the log file, mentioning that the list cannot be found.

HTTP Lua Parser Options File

Caution: RSA strongly suggests that you **do not subscribe** to the options file. Subsequent downloads of this file will overwrite all changes that you have made to the file.

Note the following:

- If you deploy the options file, it can be found in the same directory as parsers:
`/etc/netwitness/ng/parsers/.`
- The parser is not dependent upon the options file. The parser will load and run even in the absence of the options file. The options file is only required if you need to change the default settings.
- If you do not have an options file (or if your options file is invalid), the parser uses the default settings.

Note: The parser will never use both the defaults and customized options. If the options file exists and its contents can be loaded, then the defaults will not be used at all.

The `HTTP_lua_options` file contains the following options for controlling the parser:

- `registerURL`
- `splitQuery`
- `useOrigIP`
- `refererPath`
- `userAgent`
- `respReason`
- `decompress`
- `advanced`
- `customHeaders`

To change an option from **false** to **true**, edit the line inside the corresponding function, from

```
return false
to
return true
```

And similarly to go from **true** to **false**.

Note: Modifying any of these options requires a service restart to take effect; a simple parser reload is insufficient.

registerURL

Default value: **false**

Default behavior is for the host, directory, filename, extension, and query to be registered as separate meta values with the appropriate keys:

- **alias.host:** www.example.com
- **directory:** /someDir/
- **filename:** somefile.html
- **extension:** html
- **query:** ?foo=bar

This option instead registers them as a single meta value (individual keys will not be registered due to redundancy): url: www.example.com/someDir/someFile.html?foo=bar

Note: The registered URL is a reconstructed approximation: it may not be the exact URL that was clicked on.

splitQuery

Default value: **false**

Default behavior is for the entire query string from a request to be registered as a single meta value:

```
query: alpha=one&beta=two&gamma=three
```

If this option is enabled, then each element of a query string will be registered as individual meta values:

```
query: alpha=one
query: beta=two
query: gamma=three
```

useOrigIP

Default value: **true**

Default behavior is to register values from x-forwarded-for headers and the like with index key **orig_ip**. If this option is disabled, then values are registered as following:

- hostnames: "alias.host"
- IPv4: "alias.ip"
- IPv6: "alias.ipv6"
- email address: "email"
- other: "alias.host"

referrerPath

Default value: **false**

Default behavior is to register the value of a "Referer:" header as "referrer" meta. If this option is enabled, then the host, directory, filename, extension, and querystring values are broken out from Referer and registered individually. In order to avoid duplication, the entire Referer value will not be registered.

For example, assume the following header:

```
Referer: http://www.example.com/hello/world.html?foo=bar&one=two
```

If this option is disabled (default), then the following meta is registered:

```
Referer: http://www.example.com/hello/world.html?foo=bar&one=two
```

If enabled, then the following meta is registered:

```
alias.host: www.example.com
directory: /hello/
filename: world.html
extension: html
query: foo=bar&one=two
```

Note that if the **Split Query String** option is also enabled, then the query string is registered individually (see [splitQuery](#) above).

userAgent

Default value: **client**

Default behavior is to register the value of User-Agent headers with the **client** index key.

Modifying this value causes User-Agent values to additionally be registered with the specified key. If the key does not already exist it will be created (normal key name restrictions apply).

Note that this results in duplication of meta. User-agent is registered to both **client** and the specified key.

respReason

Default value: **true**

For response codes other than 2xx, default behavior is to register both the status code and reason phrase together as error meta. For example: `error: 404 Not Found`.

Disabling this option (setting to false) causes only the response code to be registered. For example: `error: 404`

decompress

Default value: **0** (means that decompression will not be performed for any content type, which maximizes performance)

Decompress content-encoded HTTP responses. Encodings gzip, deflate, chunked, and br (brotli) are supported. Enabling this option provides visibility into such responses to other parsers.

Decompression incurs a performance penalty which varies depending upon the prevalence of compressed or encoded HTTP responses seen in the environment. This can be ameliorated to some extent by choosing to only decompress specific content types.

This is a bit-packed value representing the content types to decompress, where:

- 0 Decompress OFF
- 1 application/*
- 2 audio/*
- 4 font/*
- 8 image/*
- 16 message/*
- 32 model/*
- 64 text/*
- 128 video/*
- 255 Decompress ALL content types

Just add desired values together. For example:

- To decompress **application** (1) and **text** (64), use 65 (1+64)
- To decompress **application** (1), **audio** (2), and **text** (64), use 67 (1+2+64)
- To decompress **font** (4) and **image** (8), use 12 (4+8)

The default value of 0 means that decompression will not be performed for any content type, which maximizes performance. A value of 65 specifies that content-types "application" and "text" will be decompressed. This should provide a good balance of visibility and performance. To maximize visibility, a value of 255 will enable decompression of all content types.

Enabling a content-type enables all constituent sub-types. For example, "application" includes "application/octet-stream", "application/javascript", and so on.

NOTES:

- Only valid for versions 11.0 and newer. This option has no effect on versions 10.x or older as they do not have the capability to decompress encoded HTTP responses.
- Has no effect on instances of compression which are not HTTP responses, such as compressed archive files (such as ZIP and RAR), LZMA streams, and so on.
- If enabled, then 65 is the suggested value (not 255). That will decompress data that is typically most interesting to analysts (web pages, executables, javascript), without wasting resources decompressing larger files that are typically less interesting (images, audio, video, and so forth).

advanced

Default value: **false**

If this parameter is enabled, the system performs advanced analysis of HTTP characteristics. Analysis includes only the first request and first response. Meta is registered to the key **analysis.service**.

customHeaders

Provides a mechanism for customers to register HTTP header values to specified keys.

Meta registered will be in addition to—not replacement of—standard meta registration. For example, if you specify **user-agent** headers be registered to key **foo**, **user-agent** will still also be registered to **alias.host** (or **alias.ip/alias.ipv6** if appropriate).

Therefore, beware of registering too much data, which could lead to excessive duplication and thus impact performance and data retention.

Syntax:

```
["header"] = "key",
```

Where:

- **header** is the desired HTTP header in lowercase. Do not include spaces, colons, and so forth.
- **key** is the desired meta key with which to register the value of **header**

Notes:

- (missing or bad snippet)
- Keys are registered as `format="Text"`. Do not use keys indexed in other formats.

NetWitness Investigation Model

The Investigation model organizes content, with the purpose of delivering an accurate path to information security incident response. This is a hierarchical model, four levels deep.

Threat

- Attack Phase
 - Reconnaissance
 - Delivery
 - Exploit
 - Installation
 - Command and Control
 - Action on Objectives
 - Lateral Movement
 - Data Exfiltration
 - Data Sabotage
 - Denial of Service
- Malware
 - Remote Access Trojans
 - Crimeware
 - Web Shells
 - Key Loggers

Identity

- Authentication
- Authorization
- Accounting
- Behavior Analytics
 - Entity Monitoring
 - User Mapping

Assurance

- Governance
 - Active Violations
 - Enforcement
- Risk
 - Vulnerability Management
 - Organizational Hazard
 - Enterprise Intelligence
- Compliance
 - Corporate
 - Audit

Operations

- Situation Awareness
- Event Analysis
 - Application Analysis
 - Protocol Analysis
 - File Analysis
 - Flow Analysis
 - Filters
 - Log Analysis

The [Investigation Feed](#) uses this complete model. The [Live Content Search Tags](#) uses the top two levels of this model.

Model Hierarchy

The objective of each category is to catalog existing and upcoming content with an Incident Response service-based approach. This is done to maintain mission critical revenue, protect customer data and branding, as well as to drive information security programs forward in response maturity.

This content strategy focuses on highlighting key pieces of information within an Enterprise network and log capturing environment and strives to make this data readily available for consumption and dissemination. This model aims to aid in the discovery of the preliminary and often responsive data mining tasks related to information security services.

Threat

The Threat category accounts for content that may directly lead to security incident investigations when observed on a high value corporate asset or target.

- [Attack Phase](#)
 - [Reconnaissance](#)
 - [Delivery](#)
 - [Exploit](#)
 - [Installation](#)
 - [Command and Control](#)
 - [Action on Objectives](#)
 - [Lateral Movement](#)
 - [Data Exfiltration](#)
 - [Data Sabotage](#)
 - [Denial of Service](#)
- [Malware](#)
 - [Remote Access Trojans](#)
 - [Crimeware](#)
 - [Web Shells](#)
 - [Key Loggers](#)

Examples

Rule	Categorized with meta keys
RDP Traffic Same Source to Multiple Destinations	Investigation Category (inv.category) = "Threat" Investigation Context (inv.context) = "Attack Phase" Investigation Context (inv.context) = "Action on Objectives" Investigation Context (inv.context) = "Lateral Movement"

If the source host in question is identified as being compromised, we can apply this activity to an attack phase describing pivoting within an environment.

Rule	Categorized with meta keys
Zusy Botnet	Investigation Category (inv.category) = "Threat" Investigation Context (inv.context) = "Malware" Investigation Context (inv.context) = "Remote Access Trojan" And: Investigation Category (inv.category) = "Threat" Investigation Context (inv.context) = "Attack Phase" Investigation Context (inv.context) = "Command and Control"

The distinction in specific malware situates this content in the family-type malware subtag.

Attack Phase

The intent of attack phase content is to assist incident response practitioners with the escalation, remediation or classification of observed indicators of compromise activity. This content makes use of organized threat intelligence and provides a template for incident response operations tasked at monitoring and detecting indicators in a given dataset. The attack phases are a procedural set of stages that can be carried out in a variety of ways and over a long period. An example of attack phase content would be a rule that leverages any of the stages listed below.

- **Reconnaissance:** attacker attempts to gain information about the target before the attack begins.
- **Delivery:** the attacker sends the malicious code to the victim.
- **Exploitation:** the actual execution of the exploit (only relevant when the attacker uses an exploit).
- **Installation:** the installation of malware onto the affected computer.
- **Command & Control (aka "C2"):** the attacker creates a command and control channel in order to continue to operate internal assets remotely.
- **Action on Objectives:** the attacker performs the steps to achieve actual goals inside the victim's network. Possible sub-values are as follows:
 - Enumeration
 - Lateral Movement:
 - Data Exfiltration:
 - Data Sabotage:
 - Denial of Service (aka "DDoS"):

Malware

While some known malicious behavior can be attributed to actors that have high-end intelligence, suspicious behavior can be classified independently of threat portal intelligence. Taking this approach eliminates the downside of associating a certain behavior to one actor only and better permits NetWitness Suite to detect new threats or TTP not already defined.

Breaking malware into certain family types helps content engineers to easily update any parsers or application rules that may reference a new malware variant. Response priorities can differ amongst organizations in relation to the diversity of malware types.

- **Remote Access Trojans:** Remote Access Trojans are an obvious indication that your network is actively compromised. ASOC Analytical Services recommends high response priorities be set when a remote access Trojan signature is generated
- **Crimeware:** Crimeware is currently a huge part of the Internet. Thus, it deserves a place in a response program's standard operational intake and remediation efforts. Heightened awareness on this malware type would be logical within organizations saturated with confidential identity and financial-based information.
- **Web Shells:** Web shells are an obvious indication there is active compromise. Content related to web shells should be highlighted and escalated with a sense of urgency.
- **Key Loggers:** Key loggers can be packaged up in exploit kits or delivered via spyware. Key logger content can take the form of intrusion detection signature identification numbers, or antivirus filename matches based off Virus Total analysis. Examples are aimed at identifying key loggers that do not solely utilize packet capture.

Identity

The Identity category accounts for content used in the identification or mapping of users and entities.

Identity content data is heavily utilized in response operations for validation of a particular IP address and associated end user. It classifies specific evidentiary logs to assist in incident response, and provides a means to create a baseline for NetWitness Suite.

- Authentication
- Authorization
- Accounting
- Behavior Analytics
 - Entity Monitoring
 - User Mapping

Examples

Rule	Categorized with meta keys
Account Created	Investigation Category (inv.category) = "Identity" Investigation Context (inv.context) = "Authorization" And: Investigation Category (inv.category) = "Identity" Investigation Context (inv.context) = "Assurance" Investigation Context (inv.context) = "Compliance" Investigation Context (inv.context) = "Audit"

This rule is used to highlight any accounts created in a collection. The data represented in the escalation can be intelligence that is utilized during the corporate auditing process.

Rule	Categorized with meta keys
Logon Success	Investigation Category (inv.category) = "Identity" Investigation Context (inv.context) = "Authentication" And: Investigation Category (inv.category) = "Identity" Investigation Context (inv.context) = "Assurance" Investigation Context (inv.context) = "Compliance" Investigation Context (inv.context) = "Audit"

This rule is used to highlight any successful account authentications in a collection. The data represented in this content can also be utilized during the auditing process.

Rule	Categorized with meta keys
Multiple Account Lockouts From Same or Different Users	Investigation Category (inv.category) = "Identity" Investigation Context (inv.context) = "Authorization"

Detects multiple account lockouts reported for a single or multiple users within a given time window.

Authentication

Authentication is the act of giving a user access to secure systems based on user credentials that imply authenticity. For example, the act of logging onto a system. The ways in which someone may be authenticated fall into three tags, based on what are known as the factors of authentication: something the user knows, something the user has, and something the user is.

Authorization

The process of enforcing policy as in available activities, resources, services or overall user access. For example, elevated privilege, password modifications, distribution lists, and logout activity.

Rule	Categorized with meta keys
Windows account disabled	Investigation Category (inv.category) = "Identity" Investigation Context (inv.context) = "Authorization"

The above rule tags all accounts disabled in a windows collection.

Accounting

The accounting tag contains content that measures resources utilized by a given user. This can include any of the following:

- the amount of data a user has transferred during a session
- remote access session information
- file share activity
- the overall audit of a user footprint

Rule	Categorized with meta keys
IP Profiling	Investigation Category (inv.category) = "Identity" Investigation Context (inv.context) = "Accounting"

The above content summarizes activity on a network based on a list of source IP addresses. The report includes bandwidth utilization, risk alerts, threats, top destinations, OS types, browsers and clients

Behavior Analytics

The goal of behavior analytics is to determine base lines in user behavior. Base lines are necessary for determining abnormality within overall network utilization. Behavior Analytical content helps to promote data science and user- and entity-based analytics. Within NetWitness Suite, a variety of content has already been utilized in ESA proof of concepts, which visualized host and user relationships based on MAC address, IPv4, hostname and username data from specific windows, DHCP and VPN logs.

- **Entity Monitoring:** An example of content in this subgroup could be the monitoring of the various types of administrator accounts on a corporate network. Many organizations have exclusive user names for these elevated accounts, or bind them to an associated group within active directory. Collecting, cataloging and embedding this enterprise intelligence within the content model is essential in maintaining a security program.
- **User Mapping:** Between employees adding their own devices to a network, and having multiple users on a machine, asset management can quickly become a difficult task for analysts. Any data that represents entity correlation can be stored and cataloged in this

subgroup. The underlying function of this content is to associate and identify end users while they access the network.

Assurance

The Assurance category contains content that:

- determines the corporate security posture (governance)
- adheres to internal and external audits (compliance)
- manages overall risk within the enterprise (risk)

The governance, risk, and compliance aspects of security are paramount in enabling customers to alleviate and fulfill industry requirements.

- Governance
 - Active Violations
 - Enforcement
- Risk
 - Vulnerability Management
 - Organizational Hazard
 - Enterprise Intelligence
- Compliance
 - Corporate
 - Audit

Examples

Rule	Categorized with meta keys
BYOD Mobile Web Agent Detected	Investigation Category (inv.category) = "Assurance" Investigation Context (inv.context) = "Risk" Investigation Context (inv.context) = "Compliance" Investigation Context (inv.context) = "Corporate"

The above rule detects web browsing agents not issued during standard corporate deployments, adding to the summary of overall risk exposure.

Rule	Categorized with meta keys
Proxy Anonymous Services	Investigation Category (inv.category) = "Assurance" Investigation Context (inv.context) = "Risk" Investigation Context (inv.context) = "Organizational Hazard"

The above rule detects the use of common proxy services, by using a list of domains matched against alias host.

Rule	Categorized with meta keys
Config Changes	Investigation Category (inv.category) = "Assurance" Investigation Context (inv.context) = "Compliance" Investigation Context (inv.context) = "Audit"

The above rule highlights any modifications to infrastructure in support of Compliance reporting which may assist during audit.

Governance

Governance content spawns an action for the associated escalation contact. Incident Management is the determined mindset of this category. Content that enables an act of remediation, or the education an end user is in this subgroup. An example of this could be corporate misuse of an asset, which can be carved from an organization's acceptable use or information security policies.

This subgroup identifies behavior that violates an agreement, either employee or corporate.

- **Active Violations:** Misuse of corporate resources. For example, attempting to access a specific category that is blocked. Or, HR violations based on a data loss prevention escalation. Violations often warrant an action by the security analyst or response team escalations to a risk-based organization.
- **Enforcement:** Enforcement is designated for content used in the identification of expected nefarious or delinquent behavior outside of known policies. This subgroup identifies the behavior that an incident response team should consider in remediation of a given violation. This could be in the form of user education or investigation escalation.

For example, the release of an enterprise confidential e-mail to someone outside of the Corporation based on Human Resources or a legal team's request.

Risk

Risk is defined as intelligence one may discover about the Enterprise, which may be useful within the incident response life-cycle. This can be escalation contact information, server specifics, or business intelligence.

Rule	Categorized with meta keys
Shadow IT User	Investigation Category (inv.category) = "Assurance" Investigation Context (inv.context) = "Risk"

The above content reports on suspected shadow IT usage within the organization.

- **Vulnerability Management:** Content such as vulnerable pieces of software, or updates on patches.

- **Organizational Hazard:** This subgroup consists of content that can be attributed to potentially enabling a compromise. For example, this could be passwords stored in a plain text file on the desktop of a shared administrator server. Further examples might include users using the same logon/FOB; this is an operational security failure. Additional content that would contribute to organizational hazard are instances of shadow IT.
- **Enterprise Intelligence:** Content that assesses your network and business processes. The more you know about the business itself, the better suited you are to defend it. This content could be related to high value assets or targets, or simply security infrastructure.

Compliance

The Compliance tag contains Information that may be subject to audit, or content that may contribute to readily accessing answers to important questions. Content here promotes information transparency about the security program. More importantly, the compliance section can be considered as information which supports Incident Response. Content examples can include confirming whether systems adhere to corporate compliance via installed security controls.

Rule	Categorized with meta keys
Access to Compliance Data	Investigation Category (inv.category) = "Assurance" Investigation Context (inv.context) = "Compliance"

This content is used to report on any activity related to the handling of sensitive data or restricted hosts.

- **Corporate:** content related to determining a fully IT-compliant entity or user, and required in a successful security program. This is also a task that analysts take in response to disreputable activity. In any instance of escalation, one of the first items to determine is if the security controls in place have failed.

This type of content can be used to drive security programs forward, more intelligently occupy IT-security budgets, and enable more rapid incident response.

- **Audit:** content that concerns reporting based on industry standards related to fiduciary auditing compliance (at both the state and federal level).

Operations

The Operations tag accounts for content used to aid in systematic analysis of enterprise data. This could be in the form of daily reports, dashboard visuals or the management lifecycle of customer-specific data. Most importantly, it is the content which constitutes the initial inspection of log and session collections. This is important content to deliver to an analyst because it is a prerequisite for understanding situational context.

- [Situation Awareness](#)
- [Event Analysis](#)
 - [Application Analysis](#)
 - [Protocol Analysis](#)
 - [File Analysis](#)
 - [Flow Analysis](#)
 - [Filters](#)
 - [Log Analysis](#)

Examples

Rule	Categorized with meta keys
Only ACK Flag Set in Session Containing Payload	Investigation Category (inv.category) = "Operations" Investigation Context (inv.context) = "Event Analysis" Investigation Context (inv.context) = "Protocol Analysis"

Above rule alerts when a session containing payloads have only ACK flag set.

Rule	Categorized with meta keys
NGINX HTTP Server	Investigation Category (inv.category) = "Operations" Investigation Context (inv.context) = "Event Analysis" Investigation Context (inv.context) = "Application Analysis"

The above rule detects web servers running NGINX, which is often used for malicious purposes.

Rule	Categorized with meta keys
Attachment Overload	Investigation Category (inv.category) = "Operations" Investigation Context (inv.context) = "Event Analysis" Investigation Context (inv.context) = "File Analysis"

The above rule looks for more than 4 attachments in a single session.

Situation Awareness

Comprehensive cyber situation awareness involves three key areas: computing and network components, threat information, and mission dependencies. It has added a new dimension of required awareness to traditional business operations. With this awareness, negative situations can be recognized and managed as they occur. Examples of this type of content can be daily reports and charts for visualizing certain aspects of a collection.

Event Analysis

The event analysis tag is used to classify a majority of the deep packet inspection content available within Live. Alone, a piece of content here might not lead to an active investigation. However, in combination with additional indicators of compromise the collections may require immediate review. Non-standard and service-based analysis content resides in this tag.

This tag contains the following:

- **Application Analysis:** content used to identify applications.
- **Protocol Analysis:** content used to identify anomalous sessions and deviations from standards
- **File Analysis:** content focused on the ability of NetWitness Suite to inspect files and escalate based on irregular behavior.
- **Flow Analysis:** content classified based on directionality rules:
 - Outbound Communication with the Internet,
 - Inbound Web Application Communication,
 - Intra- and Inter-DMZ communications,
 - DMZ to Inside Communications,
 - Inside to Inside Communications,
 - B2B or Partner Communications,
 - Inbound SMTP Communications,
 - Inbound Other Applications,
 - Cleartext side of Inbound VPN Connections
- **Filters:** content labeled as noise, and therefore not stored in the index of an active collection.
- **Log Analysis:** content used for log analysis such RSA Log Devices and RSA Log Collectors.

LDAP Parser Options File

Caution: RSA strongly suggests that you **do not subscribe** to the options file. Subsequent downloads of this file will overwrite all changes that you have made to the file.

Note the following:

- If you deploy the options file, it can be found in the same directory as parsers:
`/etc/netwitness/ng/parsers/.`
- The parser is not dependent upon the options file. The parser will load and run even in the absence of the options file. The options file is only required if you need to change the default settings.
- If you do not have an options file (or if your options file is invalid), the parser uses the default settings.

Note: The parser will never use both the defaults and customized options. If the options file exists and its contents can be loaded, then the defaults will not be used at all.

By default, the LDAP parse only parses the username and password if the port is 389. The options file allows parsing from any specified port, as well as some other configuration.

The `LDAP_options` file contains the following options for controlling the parser:

- **ports**
- **idOnly**
- **parseResponses**

To change an option from **false** to **true**, edit the line inside the corresponding function, from

```
return false
to
return true
```

And similarly to go from **true** to **false**.

ports

Default value: **"389, 686"**

Specifies the ports on which to look for LDAP sessions. The value should be a comma or space delineated list of port numbers. For example:

```
"389, 686"
```

LDAP sessions on ports other than those listed will not be identified, nor parsed.

idOnly

Default value: **false**

If enabled, LDAP sessions will be identified but no meta will be extracted. This may improve overall system performance.

If enabled, the **parseResponses** option is ignored.

Note: Modifying this option requires a service restart to take effect; a simple parser reload is insufficient.

parseResponses

Default value: **false**

By default, meta is not extracted from LDAP responses. If you enable this option (set its value to **true**), meta will be extracted from LDAP responses. Note that this will result in a greater amount of meta, and may decrease overall system performance.

If enabled, the meta goes to **ldap.response**, which is a non-standard key.

Log Parser Customization

On occasion, you may need to modify one or more of your log parsers. For example, you may need to fix an unknown message, or to parse certain fields differently than in the manner provided by default.

Log Parser Customization allows you to add new parser elements or modify existing ones. All customizations reside in a separate file that does not get removed or overwritten by Log Decoder upgrades or the updating parsers through the RSA Live.

Note: This feature is only available in 10.6.5 and later (including NetWitness 11.x)

Loading Order

The default parser file is loaded before the custom file (if a custom file exists). This allows users to override elements, as shown in the examples below that modify items that exist in the default file.

File Location and Naming

Log parser files are located on the Log Decoder in the following path:

```
/etc/netwitness/ng/envision/etc/devices
```

Each log parser has its own sub-folder. For example, the `ciscoasa` parser files are in the following folder:

```
/etc/netwitness/ng/envision/etc/devices/ciscoasa
```

Custom log parser files are located in the same folder as the corresponding system-provided files. For naming, you use the name of the XML file, followed by **-custom.xml**

For example, the `ciscoasa` parser consists of two files: `ciscoasa.ini` and `v20_ciscoasamsg.xml`. If you create a custom file, you need to name it `v20_ciscoasamsg-custom.xml`, and add it to the same folder, `/etc/netwitness/ng/envision/etc/devices/ciscoasa`.

Header and Message Duplication

When you customize a parser, make sure to duplicate only those headers and messages that you want to customize. That is, we recommend that you do not simply copy everything from the default parser file and then paste it into your custom XML file. Also, note that if you duplicate headers and messages that exist in the default parser, you will not be using the default versions, even if RSA updates them in the future.

Examples

The following sections contain examples for adding or modifying portions of a log parser.

All the examples use the Oracle Access Manager (oracleam) log parser.

Example Code

Cod examples are broken down into two areas:

- [Add a New Item](#)
- [Modify an Existing Item](#)

Additionally, [insertBefore and insertAfter](#) describes the usage of the **insertBefore** and **insertAfter** commands, for use when adding a new item.

Common Steps

The common steps, which are the same in all of the examples, are as follows:

1. Use an SSH tool, such as WinSCP, to navigate to the following folder on your Log Decoder:

```
/etc/netwitness/ng/envision/etc/devices/oracleam
```

2. Copy the **oracleammsg.xml** file to your local system.
3. Note the Device Messages and Version information, which comprise the first several lines of the **oracleammsg.xml** file. You need to copy these lines into your custom parser file.

```
<DEVICEMESSAGES
    name="oracleam"
    displayname="Oracle Access Manager"
    group="Access Control">

<VERSION
xml="60"
checksum="110c39794680bdeffabb5a73339d38eb"
revision="104"
device="2.0"/>
```

4. Using a text editor, create a file named **oracleammsg-custom.xml**, and add custom text, after the introductory text specified in the previous step. **The specific custom text is supplied in each of the following examples.**
5. Save the custom file as **oracleammsg-custom.xml**, and using your SSH tool, upload it to `/etc/netwitness/ng/envision/etc/devices/oracleam` on your Log Decoder.

Add a New Item

When you add an item, you use a new identifier, and optionally, an **insertBefore** or **insertAfter** command.

You can add any of the following items:

- [Add New Header](#)
- [Add New Message](#)
- [Add New Valuemap](#)
- [Add New Tagval](#)

Add New Header

Using a text editor, create a file named **oracleammsg-custom.xml**, and add the following text:

```
<DEVICEMESSAGES
    name="oracleam"
    displayname="Oracle Access Manager"
    group="Access Control">

<VERSION
xml="60"
checksum="110c39794680bdedfabb5a73339d38eb"
revision="104"
device="2.0"/>
<!-- VERSION info copied from oracleammsg.xml -->
<HEADER

    id1="0044"
    id2="0044"
    insertBefore="0005"
    content="%ORACLEAM-&lt;hfld1&gt;;: &lt;hdate&gt; &lt;htime&gt;
*&lt;htimezone&gt; - &lt;messageid&gt; &lt;!payload:messageid&gt;" />

</DEVICEMESSAGES>
```

Note the **insertBefore="0005"** line. This instructs the system to insert the new header before existing header number 0005.

Add New Message

Using a text editor, create a file named **oracleammsg-custom.xml**, and add the following text:

```
<DEVICEMESSAGES
    name="oracleam"
    displayname="Oracle Access Manager"
    group="Access Control">
```

```

<VERSION
xml="60"
checksum="110c39794680bdebfabb5a73339d38eb"
revision="104"
device="2.0"/>
<!-- VERSION info copied from oracleammsg.xml -->

<MESSAGE

    id1="AUTHZ_SUCCESS:03"
    id2="AUTHZ_SUCCESS"
    eventcategory="1302000000"
    insertAfter="AUTHZ_SUCCESS:01"
    functions="&lt;@ec_theme:Authentication&gt;&lt;@ec_
outcome:Success&gt;&lt;@event_time:*EVNTTIME($HDR,'%G/%F/%W
%N:%U:%O',hdate,htime)&gt;&lt;@msg:*PARMVAL($MSG)&gt;&lt;@:*SYSVAL
($MSGID,$ID1)&gt;";
    content="&lt;event_type&gt; - &lt;web_method&gt; -
&lt;hostname&gt;&lt;fld1&gt;- &lt;saddr&gt; - {&lt;web_
domain&gt;%&lt;fld27&gt;&lt;fld2&gt;|&lt;url&gt;&lt;fld2&gt;} -
cn=&lt;username&gt;,&lt;fld3&gt; - &lt;fld4&gt; - &lt;protocol&gt; -
&lt;obj_type&gt;&lt;fld6&gt; - &lt;context&gt; - &lt;id&gt; -
cn=&lt;fld7&gt;;cn1=&lt;fld23&gt;; uid=&lt;uid&gt;"; />

</DEVICEMESSAGES>

```

Note the **insertAfter="AUTHZ_SUCCESS:01"** line. This instructs the system to insert the new message after existing message with ID **AUTHZ_SUCCESS:01**.

Add New Valuemap

For the remaining examples, the introductory lines are not included. Add the following code after the introductory VERSION information.

```

<VALUEMAP

    name="getDisposition"
    default="$NONE"
    keyvaluepairs="0=&apos;Failure&apos;;1=&apos;Success&apos;"; />

<MESSAGE

id1="AUTHZ_SUCCESS:03"
id2="AUTHZ_SUCCESS"
eventcategory="1302000000"
insertBefore="AUTHZ_SUCCESS:01"
functions="&lt;@ec_theme:Authentication&gt;&lt;@ec_outcome:Success&gt;&lt;@event_
time:*EVNTTIME($HDR,'%G/%F/%W %N:%U:%O',hdate,htime)&gt;&lt;@msg:*PARMVAL
($MSG)&gt;&lt;@:*SYSVAL($MSGID,$ID1)&gt;";
content="&lt;event_type&gt; - &lt;web_method&gt; - &lt;hostname&gt;&lt;fld1&gt;-
&lt;saddr&gt; - {&lt;web_
domain&gt;%&lt;fld27&gt;&lt;fld2&gt;|&lt;url&gt;&lt;fld2&gt;} -
cn=&lt;username&gt;,&lt;fld3&gt; - &lt;fld4&gt; - &lt;protocol&gt; - &lt;obj_
type&gt;&lt;fld6&gt; - &lt;context&gt; - &lt;id&gt; -
cn=&lt;fld7&gt;;cn1=&lt;fld23&gt;; uid=&lt;uid&gt;"; />

```



```
</DEVICEMESSAGES>
```

Add New Tagval

Add the following code after the introductory VERSION information.

```
<TAGVALMAP
```

```
    pairdelimiter="^^" encapsulator="&quot;" />
```

```
<VALUEMAP
```

```
    name="getDisposition"
    default="$NONE"
    keyvaluepairs="0=&apos;Failure&apos;;1=&apos;Success&apos; " />
```

```
<MESSAGE
```

```
    id1="ORACLEAM_TVM"
    id2="ORACLEAM_TVM"
    eventcategory="1901000000"
    tagval="true"
    missField="true"
    functions="&lt;@msg:*PARMVAL($MSG)&gt;&lt;@event_time:*EVNTTIME
($MSG, '%W-%G-%F %H:%T:%S', fld3)&gt;&lt;@disposition:*getDisposition
(fld12)&gt;&lt;@msg_id:*PARMVAL(event_type)&gt;&lt;@vid:*PARMVAL
(event_type)&gt;&lt;@event_id:*STRCAT(event_
type,_, disposition)&gt;&lt;@event_cat:*getEventLegacyCategory(event_
id)&gt;&lt;@event_cat_name:*getEventLegacyCategoryName(event_cat)&gt;";
    content="IAU_EVENTTYPE=&lt;event_type&gt;^^IAU_
EVENTCATEGORY=&lt;category&gt;^^IAU_COMPONENTTYPE=&lt;event_
source&gt;^^IAU_HOSTID=&lt;dhost&gt;^^IAU_
HOSTNWADDR=&lt;daddr&gt;^^IAU_AGENTID=&lt;fld1&gt;^^IAU_
PROCESSID=&lt;process_id&gt;^^IAU_SESSIONID=&lt;sessionid&gt;^^IAU_
SSOSESSIONID=&lt;sessionid1&gt;^^IAU_
APPLICATIONNAME=&lt;application&gt;^^IAU_
APPLICATIONDOMAINNAME=&lt;fld2&gt;^^IAU_
EVENTSTATUS=&lt;fld12&gt;^^IAU_TSTZORIGINATING=&lt;fld3&gt;^^IAU_
THREADID=&lt;fld4&gt;^^IAU_INITIATOR=&lt;username&gt;^^IAU_
USERID=&lt;uid&gt;^^IAU_MESSAGETEXT=&lt;event_description&gt;^^IAU_
REMOTEIP=&lt;saddr&gt;^^IAU_RESOURCE=&lt;fld5&gt;^^IAU_
DOMAINNAME=&lt;domain&gt;^^IAU_SERVERNAME=&lt;hostname&gt;^^IAU_
INSTANCENAME=&lt;instance&gt;^^IAU_AUTHORIZATIONPOLICYID=&lt;policy_
id&gt;^^IAU_AUTHENTICATIONPOLICYID=&lt;policy_id&gt;^^IAU_
RESOURCEHOST=&lt;shost&gt;^^IAU_RESOURCEURI=&lt;url&gt;^^IAU_
ADDITIONALINFO=&lt;fld7&gt; " />
```

```
</DEVICEMESSAGES>
```

Note the **tagval="true"** code in the message. We are adding this message that uses the new Tagval map.

insertBefore and insertAfter

As shown in some of the previous examples, the **insertBefore** and **insertAfter** commands instruct the system about where to place the new items when combining the standard and custom XML definition files, as it creates a unified parser during processing.

Note: If both **insertBefore** and **insertAfter** are defined, **insertBefore** will be used, and a warning will be logged. If neither is specified, the header or message is added at the end of the combined parser definition.

Modify an Existing Item

To modify an existing element, you use the same identifiers as an existing item, and change the contents. See the examples to modify any of the following items:

- [Modify Header](#)
- [Modify Message](#)
- [Modify Valuemap](#)
- [Modify Tagval](#)

Modify Header

This example replaces the Header that has an ID of 0004. Add the following code after the introductory VERSION information.

```
<HEADER
  id1="0004"
  id2="0004"
  content="%ORACLEAM-&lt;hfld1&gt;;: &lt;hdate&gt; &lt;htime&gt;
  *&lt;htimezone&gt; - &lt;messageid&gt; &lt;!payload:messageid&gt;" />
</DEVICEMESSAGES>
```

Modify Message

This example replaces the Message that has an ID of AUTHZ_SUCCESS:01. Add the following code after the introductory VERSION information.

```
<MESSAGE
  id1="AUTHZ_SUCCESS:01"
  id2="AUTHZ_SUCCESS"
  eventcategory="1302000000"
  functions="&lt;@ec_theme:Authentication&gt;&lt;@ec_
  outcome:Success&gt;&lt;@event_time:*EVNTTIME($HDR, '%G/%F/%W
```

```
%N:%U:%O',hdate,htime)&gt;&lt;@msg:*PARMVAL($MSG)&gt;&lt;@:*SYSVAL
($MSGID,$ID1)&gt;"
content="&lt;event_type&gt; - &lt;web_method&gt; -
&lt;hostname&gt;&lt;fld1&gt;- &lt;saddr&gt; - {&lt;web_
domain&gt;%&lt;fld27&gt;&lt;fld2&gt;|&lt;url&gt;&lt;fld2&gt;} -
cn=&lt;username&gt;,&lt;fld3&gt; - &lt;fld4&gt; - &lt;protocol&gt; -
&lt;obj_type&gt;&lt;fld6&gt; - &lt;context&gt; - &lt;id&gt; -
cn=&lt;fld7&gt;,&lt;fld23&gt;,&lt;uid&gt;" />
</DEVICEMESSAGES>
```

Modify Valuemap

This example replaces the **getDisposition** Valuemap. Add the following code after the introductory VERSION information.

```
<VALUEMAP
    name="getDisposition"
    default="$NONE"
    keyvaluepairs="0=&apos;Failure&apos;|1=&apos;Success&apos;|3=&apos;Tes
t&apos;" />
</DEVICEMESSAGES>
```

In this example, we are assuming the device XML, **oracleammmsg.xml**, includes a Valuemap named **getDisposition**, and that we are changing the existing information, for example we might be adding a new key value pair, **3='Test'**.

Modify Tagval

This example replaces the existing Tagval. Add the following code after the introductory VERSION information.

```
<TAGVALMAP
    pairedlimiter="^^^" encapsulator="&quot;" />
<MESSAGE
    id1="ORACLEAM_TVM"
    id2="ORACLEAM_TVM"
    eventcategory="1901000000"
    tagval="true"
    missField="true"
    functions="&lt;@msg:*PARMVAL($MSG)&gt;&lt;@event_time:*EVNTTIME
($MSG, '%W-%G-%F %H:%T:%S', fld3)&gt;&lt;@disposition:*getDisposition
(fld12)&gt;&lt;@msg_id:*PARMVAL(event_type)&gt;&lt;@vid:*PARMVAL
(event_type)&gt;&lt;@event_id:*STRCAT(event_
type,_,disposition)&gt;&lt;@event_cat:*getEventLegacyCategory(event_
id)&gt;&lt;@event_cat_name:*getEventLegacyCategoryName(event_cat)&gt;"
content="IAU_EVENTTYPE=&lt;event_type&gt;^^^IAU_
EVENTCATEGORY=&lt;category&gt;^^^IAU_COMPONENTTYPE=&lt;event_
source&gt;^^^IAU_HOSTID=&lt;dhost&gt;^^^IAU_
```

```
HOSTNWADDR=&lt;fld1&gt;;^^^IAU_AGENTID=&lt;fld1&gt;;^^^IAU_
PROCESSID=&lt;process_id&gt;;^^^IAU_SESSIONID=&lt;sessionid&gt;;^^^IAU_
SSOSESSIONID=&lt;sessionid1&gt;;^^^IAU_
APPLICATIONNAME=&lt;application&gt;;^^^IAU_
APPLICATIONDOMAINNAME=&lt;fld2&gt;;^^^IAU_
EVENTSTATUS=&lt;fld12&gt;;^^^IAU_TSTZORIGINATING=&lt;fld3&gt;;^^^IAU_
THREADID=&lt;fld4&gt;;^^^IAU_INITIATOR=&lt;username&gt;;^^^IAU_
USERID=&lt;uid&gt;;^^^IAU_MESSAGETEXT=&lt;event_description&gt;;^^^IAU_
REMOTEIP=&lt;saddr&gt;;^^^IAU_RESOURCE=&lt;fld5&gt;;^^^IAU_
DOMAINNAME=&lt;domain&gt;;^^^IAU_SERVERNAME=&lt;hostname&gt;;^^^IAU_
INSTANCENAME=&lt;instance&gt;;^^^IAU_AUTHORIZATIONPOLICYID=&lt;policy_
id&gt;;^^^IAU_AUTHENTICATIONPOLICYID=&lt;policy_id&gt;;^^^IAU_
RESOURCEHOST=&lt;shost&gt;;^^^IAU_RESOURCEURI=&lt;url&gt;;^^^IAU_
ADDITIONALINFO=&lt;fld7&gt;;" />
```

```
</DEVICEMESSAGES>
```

Lua Debugging Tool

Because I could not get this topic to publish correctly (images did not publish, plus title mishigas), I have moved it to the 11.4 Project, under **2 Config > ATD**. I will publish the topic there, move it to Staging, and then move it to Content after 11.4 GA. Going forward, if we need to update the topic, we should do that in reverse: move the topic to 11.x online space, then republish from ATD target, and then move it back to Content. Hopefully, that will allow us to keep the same URL for the topic.

Mail Lua Parser Options File

Caution: RSA strongly suggests that you **do not subscribe** to the options file. Subsequent downloads of this file will overwrite all changes that you have made to the file.

Note the following:

- If you deploy the options file, it can be found in the same directory as parsers:
`/etc/netwitness/ng/parsers/.`
- The parser is not dependent upon the options file. The parser will load and run even in the absence of the options file. The options file is only required if you need to change the default settings.
- If you do not have an options file (or if your options file is invalid), the parser uses the default settings.

Note: The parser will never use both the defaults and customized options. If the options file exists and its contents can be loaded, then the defaults will not be used at all.

The **Mail_lua_options** file contains the following options for controlling the parser:

- **registerEmailSrcDst**
- **parseQuoted**
- **registerAddressHosts**
- **parseReceived**

To change an option from **false** to **true**, edit the line inside the corresponding function, from

```
return false
to
return true
```

And similarly to go from **true** to **false**.

Note the following:

- Modifying any of these options requires a service restart to take effect; a simple parser reload is insufficient.
- (missing or bad snippet)

registerEmailSrcDst

Default value: **false**

This option determines whether to register email address meta using the index keys **email.src** and **email.dst**.

If set to **false**, all email address meta is registered with the index key **email**.

If set to **true**:

- Originating email addresses will be registered with the index key **email.src**
- Recipient email addresses will be registered with the index key **email.dst**

parseQuoted

Default value: **false**

If set to false (default) then meta will not be extracted from headers which are contained within an email message (i.e., from a quoted message).

If set to true, then headers from quoted messages will be parsed.

registerAddressHosts

Default value: **false**

This option determines whether to register the host portion of email addresses as meta. The key used to register is **alias.host**, **alias.ip**, or **alias.ipv6**, as appropriate.

parseReceived

Default value: **true**

This option determines whether to register meta from **Received:** headers.

Many mail transfer agents (MTAs) add badly formatted information into **Received:** headers. This often manifests as as **alias.host** meta that is not a hostname. If this is problematic in your environment, disable parsing of **Received:** headers by setting the value to **false**.

Packet Parsers

This topic discusses and describes the packet (Lua) parsers available in RSA NetWitness Platform. If you need a parser that does not already exist, you can [Request a Parser](#).

Note: More information on each of these parsers is available in Live. Navigate to Live search, and select **RSA Lua Parser** in the **Resource Types** field. From the results, select any parser and click to display all the information for the parser.

Context

Packet parsers identify the application layer protocol of sessions seen by the Decoder, and extract meta data from the packet payloads of the session.

Every packet parser is able to extract meta from every session. For example, a webmail session will be parsed by both an HTTP parser which identifies the session as HTTP and extracts meta from HTTP headers, and by a MAIL parser which extracts email-related meta from message headers. Further, if the session were to contain an executable file, its presence would be detected by a windows executable parser.

Packet parsers in RSA NetWitness Suite may be broadly classified as:

- **System or Native parsers:** These are compiled into the Decoder base code. Updates are delivered along with updates to RSA NetWitness Suite. Many system parsers have Lua equivalents. In these cases, generally, the native parser may perform faster, while the Lua parser may extract more meta.
- **Lua parsers:** these are written in the Lua programming language, and delivered via Live. Customers can write their own custom Lua parsers.
- **Flex parsers:** these were written in a proprietary scripting language, Flex, and delivered via Live. These are now discontinued, and no longer delivered in Live. Every existing Flex parser has a better Lua equivalent, and all customers using NetWitness Suite should not be using Flex parsers.

Packet Parsers in NetWitness Suite

The following table describes the Lua parsers delivered with RSA NetWitness Platform.

Parser Name	Description
apt_artifacts	Detects possible apt WMI and windows registry manipulation.
Avamar	Identifies Avamar Backup and Recovery, TCP port 28001.

Parser Name	Description
BGP_lua	Identifies BGP Routing Protocol.
bittorrent_lua	Identifies the bittorrent protocol and registers the name of the file being downloaded.
Canon_BJNP	Identifies Canon printer discover protocol BJNP.
cerber	Detects potential Cerber ransomware beaconing.
china_chopper	Detects cleartext China Chopper sessions.
creditcard_detection_lua	Attempts to detect possible credit card numbers and validate with Luhn's Algorithm.
CustomTCP	Detects CustomTCP beaconing activity. Registers C2 domain and victim hostname as alias.host meta.
db2_lua	Extracts queries from DB2 database protocol sessions.
DCERPC	Extracts action and Kerberos authentication from Microsoft's DCERPC protocol.
Derusbi_Server_Handshake	Detects Derusbi server handshake.
DHCP_lua	Identifies DHCP (BOOTP) and DHCPv6, extracts hosts and addresses.
DNP3_lua	DNP3 Distributed Network Protocol (SCADA).
DNS_verbose_lua	Identifies DNS sessions. Registers query and response records including record type. Registers protocol error messages.
dr_watson_lua	Detects Dr Watson crash report and registers name of crashed process.
duqu_lua	Detects binaries that may be related to the duqu threat.
DynDNS	Detects dynamic DNS hosts and servers.
ein_detection_lua	Attempts to detect Employer Identification Numbers.

Parser Name	Description
ethernet_oui	Determines the manufacturer of eth.
Evilgrab	Detects possible Evilgrab APT malware activity.
exif	Extract longitude and latitude coordinates from exif data embedded in JPEG files.
fingerprint_7zip	Detects 7zip archive files.
fingerprint_access_db_lua	Identifies Microsoft Access database files.
fingerprint_apple_dmg_lua	Detects Mac OS X Disk Copy Disk Image files.
fingerprint_apple_ios_lua	Detects Apple IOS App files.
fingerprint_apple_iwork_lua	Detects Apple iWork files (Pages, Numbers and Keynote).
fingerprint_appleExec_lua	Detects MAC OSX executable binary files.
fingerprint_bmp	Detects BMP format image files.
fingerprint_cab	Identifies cabinet files (cab).
fingerprint_cad_lua	Detects Autodesk Autocad DWG, DXF, and DWF files.
fingerprint_chm_lua	Identifies Microsoft Compiled Help files, and detects potentially suspicious elements within.
fingerprint_flash	Detects Adobe Flash (swf) files.

Parser Name	Description
fingerprint_font	Identifies font files: embedded opentype (eot), web open format (woff), opentype (otf), and truetype (ttf).
fingerprint_gif_lua	Identifies GIF files.
fingerprint_gzip	Detects files which have been compressed using the gzip family of compression programs (gzip, bzip, etc).
fingerprint_java	Detects Java JAR and CLASS files.
fingerprint_javascript_lua	Detect javascript, and suspicious javascript actions and anomalies.
fingerprint_job	Identifies windows job task scheduling files.
fingerprint_jpg_lua	Detects JPEG image files.
fingerprint_lnk_lua	Identifies lnk files and detects possible exploit characteristics.
fingerprint_msi_lua	Identifies Microsoft OLE / Compound Document Format Windows Installer files.
fingerprint_mssql_lua	Detects Microsoft SQL Server database files.
fingerprint_office_lua	Identifies Microsoft Office 95-2007 Word, Excel, and Powerpoint documents.
fingerprint_pdf_lua	Identifies PDF files and detects risky characteristics.
fingerprint_pff	Detects Microsoft Outlook Personal File Folder objects such as pab, pst, and ost.
fingerprint_pkcs12_lua	Detects PKCS #12 format private key files.
fingerprint_png_lua	Detects PNG image files.

Parser Name	Description
Fingerprint_Private_Key	Detects SSH and PGP private key files.
fingerprint_rar_lua	Detects RAR archive files.
fingerprint_rtf_lua	Detects RTF files.
fingerprint_unix_script_lua	Identifies shell, perl, ruby, and python scripts.
fingerprint_webm	Detects webm and matroska video files.
fingerprint_zip	Detects PK format zip files, and extracts the names of files contained in the archive.
FIX_lua	Identifies the Financial Information Exchange Protocol. Form_Data_lua Extracts submitted values from HTTP POST actions.
Form_Data_lua	Extracts submitted values from HTTP POST actions.
FTP_lua	File Transfer Protocol (FTP) RFC 959.
ghost	Detects likely Ghost Rat beacon sessions.
glass_rat	Detects the network communication used by the GlassRAT Trojan identified by RSA Research.
gnutella_lua	Identifies the Gnutella file sharing protocol.
HTML_threat	Detects common HTML threat techniques such as hidden frames and embedded objects.
htran_lua	Identifies the error message generated by the htran redirection tool.
HTTP_lua	Extracts values from HTTP protocol request and response headers.
HTTP_lua_options	Use this file to influence the behavior of the HTTP_lua parser. For details, see HTTP Lua Parser Options File .

Parser Name	Description
HTTP_SQL_Injection	Detect possible injection of SQL commands in HTTP requests.
ICMP	Provides types and codes from ICMP packets.
IDN_homograph	Detects punycode-encoded internationalized domain names which use non-Latin Unicode code points whose glyphs resemble those of Latin Unicode code points. Registers the decoded homograph as analysis.service meta. Reference the RSA Link blog post from RSA Research for more details about this threat: Dissecting PunyCode - Not All Characters are Created Equal .
IMAP_lua	Identifies IMAP, registers commands, errors, usernames, and passwords.
IRC_verbose_lua	Expanded IRC parsing.
ISAKMP	Identifies ISAKMP (Internet Security Association and Key Management Protocol).
iSCSI	Identifies SCSI-over-IP.
JSON-RPC	Identifies JSON-RPC 2.0 streams. Will not identify JSON-RPC 1.0 streams, and may not identify JSON-RPC over transports such as HTTP.
Kerberos	Extracts meta from the Kerberos network protocol.
LDAP	Lightweight Directory Access Protocol, and extensions.
LDAP_options	Lightweight Directory Access Protocol, and extensions. Use this file to influence the behavior of the LDAP parser. For details, see LDAP Parser Options File .
Lync	Identifies Microsoft Lync (formerly Microsoft Office Communicator, Windows Messenger).
MAIL_lua	Extracts values from email messages, such as email addresses, subject, and client.
Mail_lua_options	Use this file to influence the behavior of the Mail_lua parser. For details, see Mail Lua Parser Options File .
Mitozhan	Detects Mitozhan malware command and control.
modbus	Identifies MODBUS TCP/IP, extracts commands, errors, and device identifications.

Parser Name	Description
MSU_rat	Detects MSU RAT activity.
NetBIOS_lua	NetBIOS over TCP/IP: NBNS, NBDS, NBSS.
NFS_lua	Identifies and parses RPC-related protocols NFS, MOUNT, and PORTMAP.
NTLMSSP_lua	Extracts Active Directory user information from NTLM HTTP headers from proxy authorization.
ntp_lua	Identifies Network Time Protocol.
OCSP_lua	Extracts certificate information and status from OCSP messages.
Packers	Detects specific packer used to pack executables.
phishing_lua	Registers the host portion from each URL found within an email.
plugx	Detect PlugX malware.
Poison_Ivy	Detects Poison Ivy RAT activity.
POP3_lua	Post Office Protocol version 3.
Proxy_Block_Page	Parses proxy denied exception pages.
pvid	Detects PGV_PVID malware activity. PGV_PVID is a cookie string the actor put into the malware's POST routine.
pwdump	Detects output from Windows password dumping tools such as pwdump.
QQ_lua	Identifies QQ (OICQ protocol) sessions.
radius	Remote Authentication Dial In User Service.
RDP_lua	Identifies the Microsoft Remote Desktop Protocol.
rekaf	Detects a variant of rekaf and derives the xor key (crypto) and name of the infected host.
ripng_lua	Identifies the RIP routing protocol.
rlogin	Identifies Remote Login protocol.
rsync	Identifies the RSYNC ;Network Protocol.
rtmp_lua	Real Time Messaging Protocol.

Parser Name	Description
RTSP	Identifies the Real Time Streaming Protocol.
SCCP_lua	Cisco Skinny Client Control Protocol.
Search_Engines	Extracts search terms from search engine queries.
sekur	Detects the initial handshake of the Sekur/Anunak Trojan.
session_analysis	Analyzes session characteristics such as bytes transmitted vs bytes received, TCP flags seen, etc.
shadyrat_lua	Identifies potential artifacts related to shadyrat command and control traffic.
Signed_Executable	Extracts the Certificate Authority, Subject, and Serial Number from the first x509v3 certificate in the certificate chain of a signed executable.
SIP_lua	Session Initiation Protocol (SIP).
SMB_lua	Parses the Microsoft SMB/CIFS protocol, versions 1 and 2.
SMTP_lua	Parses the SMTP protocol (RFC 5321).
SNMP_lua	Parses SNMP versions 1, 2c, 2p, 2u, and 3.
socks_lua	Identifies Socks protocol version 4 and 5.
SoulSeek_lua	Identifies the SoulSeek file sharing protocol.
spectrum_lua	Determines which sessions are sent to Malware Analysis, based upon file types seen in the session, and total session size.
SSH_lua	Identifies SSH protocol.
struts_exploit	Detects a possible Remote Code Execution attack when using the Struts REST plugin with XStream handler to handle XML payloads.
supercmd	Detects SuperCMD Trojan beaconing. For details on the SuperCMD Rat, see the SUPERCMD RAT RSA blog post .
teredo	Identifies teredo tunneled sessions. Performs identification only. No meta is extracted.
TDS_lua	Identifies Microsoft SQL Server 'Tabular Data Stream' protocol.
TFTP_lua	Identifies Trivial File Transfer Protocol, extracts names of files transferred.

Parser Name	Description
TLD_lua	Extracts the top-level domain and second-level domain portions from hostnames.
TLD_lua_options	Use this file to influence the behavior of the TLD_lua parser. For details, see TLD Lua Parser Options File .
TLS_lua	Identifies SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1, and TLS 1.2.
TN3270E_lua	Identifies IBM TN3270E sessions.
traffic_flow	Provides subnet names for internal networks, and directionality of the session (inbound, outbound, lateral).
traffic_flow_options	This is an optional file for use with the traffic_flow Lua parser. If used, this file provides a way for customers to configure internal subnets as described within the full product documentation for this parser. For details, see Editing the Options File in the Traffic Flow Lua topic.
vCard_lua	Extracts fullname and email values from vCard, xCard, jCard, and hCard formats.
VNC	Identifies the Remote Framebuffer protocol used by VNC and its derivatives.
windows_command_shell_lua	Identifies Microsoft Windows command shell sessions.
windows_executable	Identifies windows executables, and analyzes them for anomalies and other suspicious characteristics.
X11_lua	Identifies the X11 protocol (RFC 1013).
xor_executable_lua	Detects executables that have been xor or hex encoded.

Discontinued Packet Parsers

The following table lists the Lua parsers that have been removed from the system.

Name	Description	Notes
AIM_ lua	OSCAR protocol used by AIM (AOL Instant Messenger) and ICQ, and AIM-express web client.	As of December 15, 2017, AOL Instant Messenger products and services have been shut down and no longer work.
BITS	Identifies Microsoft BITS Protocol.	BITS was added to HTTP_lua, making the standalone BITS parser redundant. BITS parsing in HTTP_lua is also much more complete than it was in the standalone parser.

Phishing Lua Parser Options File

Caution: RSA strongly suggests that you **do not subscribe** to the options file. Subsequent downloads of this file will overwrite all changes that you have made to the file.

Note the following:

- If you deploy the options file, it can be found in the same directory as parsers:
`/etc/netwitness/ng/parsers/`.
- The parser is not dependent upon the options file. The parser will load and run even in the absence of the options file. The options file is only required if you need to change the default settings.
- If you do not have an options file (or if your options file is invalid), the parser uses the default settings.

Note: The parser will never use both the defaults and customized options. If the options file exists and its contents can be loaded, then the defaults will not be used at all.

The **phishing_lua_options** file contains the following options for controlling the parser:

- Deduplicate Host Registration
- Check Host Consistency
- Whitelist Domain
- Register URL Components
- Register Entire URL
- Host Key

To change an option from **false** to **true**, edit the line inside the corresponding function, from

```
return false
to
return true
```

And similarly to go from **true** to **false**.

Note: Modifying any of these options requires a service restart to take effect; a simple parser reload is insufficient.

Deduplicate Host Registration

Name: **deduplicate**. Default value: **true**

By default, if the same host portion appears in multiple HREFs within a session, it will only be registered once for that session.

If this option is disabled, then the host portion of an HREF will be registered each time it is seen, regardless of whether it has already been registered previously for that session.

Note that this option only affects the behavior of this parser. A host may still be registered by another parser. This option has no effect on the **Check Host Consistency** option.

Check Host Consistency

Name: **hostCheck**. Default value: **true**

Compares the host portions of all URLs found within an HREF. If the host portion is a hostname, then only the domain portion is compared. If the host portion is an IP address, the entire IP is compared.

Whitelist Domain

Name: **whitelistDomain**. Has no default value.

Intended for sites that rewrite HREFs in email messages. For example:

```
<a href="http://www.foo.com">http://www.foo.com</a>
```

becomes:

```
<a href="http://redirect.example.com?url=http://www.foo.com">http://www.foo.com</a>
```

This option accepts a domain to exclude from consistency checking. The domain must be enclosed in quotes, such as **"example.com"**.

Note that in the following example, an alert will still be registered even if **"example.com"** is whitelisted:

```
<a href="http://redirect.example.com?url=http://www.foo.com">http://www.bar.com</a>
```

Register URL Components

Name: **urlComponents**. Default value: **false**.

Warning: Do not enable this option if you are enabling the Register Entire URL (**registerURL**) option.

In addition to host meta, this option registers the components of each URL found. For example, assume the following URL:

```
http://www.example.com/directory/filename.ext?p=foo%3Dbar.
```

This registers the following meta:

- directory: **directory**
- filename: **filename.ext**
- extension: **ext**
- query: **p=foo%3Dbar**

No deduplication of components (other than host) is performed, even if the option **Deduplicate Host Registration** is enabled.

Register Entire URL

Name: **registerUrl**. Default value: **false**.

Warning: URLs are highly unique. Therefore, enabling this option will bloat the metadb, decreasing performance and retention, and is NOT ADVISED.

Do not enable this option if also enabling **Register URL Components**.

Registers the entirety of each URL found. The URL will be registered with the meta key **url**. Registered URLs will be a maximum of 256 characters (this is a standard meta length limitation).

No deduplication of URLs performed, even if the **Deduplicate Host Registration** option is enabled.

Host Key

Name: **hostKey**. Default is **alias.host**.

Default behavior is to register extracted hosts as **alias.host**, **alias.ip**, or **alias.ipv6** as appropriate.

Modifying this value will cause extracted hosts to instead be registered with the specified key. If the key does not already exist, it will be created. Normal key name restrictions apply.

SMTP Lua Parser Options File

Caution: RSA strongly suggests that you **do not subscribe** to the options file. Subsequent downloads of this file will overwrite all changes that you have made to the file.

Note the following:

- If you deploy the options file, it can be found in the same directory as parsers:
`/etc/netwitness/ng/parsers/`.
- The parser is not dependent upon the options file. The parser will load and run even in the absence of the options file. The options file is only required if you need to change the default settings.
- If you do not have an options file (or if your options file is invalid), the parser uses the default settings.

Note: The parser will never use both the defaults and customized options. If the options file exists and its contents can be loaded, then the defaults will not be used at all.

The `SMTP_lua_options` file contains the following options for controlling the parser:

- **registerEmailSrcDst**
- **registerAddressHosts**
- **errorCodeOnly**

To change an option from **false** to **true**, edit the line inside the corresponding function, from

```
return false
to
return true
```

And similarly to go from **true** to **false**.

registerEmailSrcDst

Default value: **false**

Determines whether or not to register email address meta using the index keys **email.src** and **email.dst**.

- If set to **false**, all email address meta is registered with the index key **email**.
- If set to **true**:

- Originating email addresses are registered with the index key **email.src**
- Recipient email addresses are registered with the index key **email.dst**
- Targets of EXPN and VRFY are registered with the index key **email.dst**

Note: Modifying this option requires a service restart to take effect; a simple parser reload is insufficient.

registerAddressHosts

Default value: **false**

Determines whether or not to register the host portion of email addresses as meta. The key used to register is **alias.host**, **alias.ip**, or **alias.ipv6** as appropriate.

errorCodeOnly

Default value: **true**

Determines whether or not to register error codes only, or the entire error messages.

- If set to **true**, only the error code (such as "450", "550", and so on) is registered.
- If set to **false**, the entire error message is registered (e.g., "550 unknown user"). In the case of a 421 server greeting, the server name is removed from the error text and be registered as **alias.[host|ip|ipv6]** instead.

System Parsers

This topic lists the native parsers available in RSA Security Analytics.

Context

Packet parsers identify the application layer protocol of sessions seen by the Decoder, and extract meta data from the packet payloads of the session.

Every packet parser is able to extract meta from every session. For example, a webmail session will be parsed by both an HTTP parser which identifies the session as HTTP and extracts meta from HTTP headers, and by a MAIL parser which extracts email-related meta from message headers. Further, if the session were to contain an executable file, its presence would be detected by a windows executable parser.

Packet parsers in RSA NetWitness Suite may be broadly classified as:

- **System or Native parsers:** These are compiled into the Decoder base code. Updates are delivered along with updates to RSA NetWitness Suite. Many system parsers have Lua equivalents. In these cases, generally, the native parser may perform faster, while the Lua parser may extract more meta.
- **Lua parsers:** these are written in the Lua programming language, and delivered via Live. Customers can write their own custom Lua parsers.
- **Flex parsers:** these were written in a proprietary scripting language, Flex, and delivered via Live. These are now discontinued, and no longer delivered in Live. Every existing Flex parser has a better Lua equivalent, and all customers using NetWitness Suite should not be using Flex parsers.

System Parsers in RSA NetWitness Platform

The following table describes the system parsers delivered with RSA NetWitness Platform.

Note For content that has been discontinued, see [Discontinued Content](#).

Name	Description
ALERTS	Alerts
DHCP	Dynamic Host Configuration Protocol

Name	Description
DNS	Domain Name Service
enVision	Log Decoder Service
FTP	File Transfer Protocol
GeoIP	Geographic data based on ip.src
GTalk	Google Talk
H323	H.323 Teleconferencing Protocol
HTTP	Hyper Text Transport Protocol
HTTPS	Secure Socket Layer Protocol
IRC	Internet Relay Chat Protocol
MAIL	Standard E-Mail Format (RFC822)
NETBIOS	NETBIOS computer name and parser
NETWORK	Network Layer parser
NFS	Network File System
NNTP	Network News Transport Protocol
PGP	PGP blocks within network traffic parser
POP3	Post Office Protocol
RIP	Routing Information Protocol
RTP	Real Time Protocol for audio/video
SCCP	Cisco Skinny Client Control Protocol
SEARCH	Searches content for keywords and/or regular expressions
SIP	Session Initiation Protocol

Name	Description
SMB	Server Message Block
SMIME	SMIME blocks within network traffic
SMTP	Simple Mail Transport Protocol
SNMP	Simple Network Management Protocol
SSH	Secure Shell
TDS	MSSQL and Sybase Database Protocol
TELNET	TELNET Protocol
TFTP	Trivial File Transfer Protocol
TNS	Oracle Database Protocol
VCARD	Extracts Full Name and E-mail information

RSA Threat Content mapping with MITRE ATT&CK™

Introduction to MITRE ATT&CK™ Navigator

Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK™) for enterprise is a framework which describes the adversarial actions or tactics from Initial Access (Exploit) to Command & Control (Maintain). ATT&CK Enterprise deals with the classification of post-compromise adversarial tactics and techniques against Windows, Linux and MacOS. This community-enriched model adds techniques used to realize each tactic. These techniques are not exhaustive, and the community adds them as they are observed and verified.

All RSA Application Rules, ESA Rules and Lua Parsers have been mapped to one or more ATT&CK™ techniques. This mapped content can be viewed in JSON format and can be graphically represented to measure ATT&CK™ matrix coverage by RSA Threat Content.

See the following blog posts on RSA Link for more information:

- For details on Threat Mapping and how to view a JSON in ATT&CK Navigator, see [Introduction to MITRE's ATT&CK™ and Mapping to ESA Rules](#).
- To learn more about content mapping, graphical representation and to access JSON files with mapped content, see [RSA Threat Content mapping with MITRE ATT&CK™](#).

Note: Multiple blog posts on RSA Link contain samples of JSON.

Generate of MITRE ATT&CK™ Metadata for RSA NetWitness Content

The Investigation Feed generates the metadata for the MITRE ATT&CK™ framework for RSA Application Rules and RSA Lua Parser logic. The keys *ATT&CK Tactic* and *ATT&CK Technique* are populated on a match to an out-of-the-box rule.

You can view populated meta in the Investigate > Navigate view.

The screenshot shows the RSA Investigate interface with the following details:

- Navigation:** Respond, Investigate (selected), Monitor, Configure, Admin
- Menu:** NAVIGATE, EVENTS, HOSTS, FILES, ENTITIES, MALWARE ANALYSIS
- Query:** endpointloghybrid - Concentrator
- Time Range:** Last 2 Days
- Results:**
 - ATT&CK Technique** [attack:technique] (20 of 20+ values)
 - hidden files and directories (78) - powershell (22) - user execution (19) - command-line interface (19) - credential dumping (14) - process injection (13) - service execution (10) - obfuscated files or information - masquerading (3) - remote system discovery (1) - new service (1) - disabling security tools (1) - create account (1) - account discovery (1) ... [show more](#)
 - ATT&CK Tactic** [attack:tactic] (7 values)
 - defense evasion (100) - execution (88) - persistence (17) - credential access (14) - privilege escalation (9) - initial access (6) - discovery (5)
 - Behaviors of Compromise** [boc] (20 of 20+ values)
 - runs service control tool (71) - unsigned writes executable to windows directory (38) - runs powershell (35) - os process runs command shell (24) - unsigned creates remote thread (20) - unsigned writes execution tool (6) ... [show more](#)
 - File Analysis** [analysis:file] (20 of 20+ values)
 - in appdata directory (176) - in hidden directory (135) - autorun unsigned hidden (72) - process authorized in firewall (71) - file hidden (32) - autorun unsigned logon type registry startup method (24) - autorun directory (8) - in root of program directory (8) - glina replacement (8) - autorun unsigned in temp directory (8) - autorun unsigned in appdata local directory (8) - autorun invalid signature windows directory
 - Session Analysis** [analysis:session] (2 values)
 - not top 20 dst (352) - watchlist dst (1)
 - Destination IP address** [ip:dst] (20 of 20+ values)
 - 192.168.134.47 (50) - 192.168.134.255 (41) - 51.143.106.177 (14) - 192.168.134.51 (6) - 204.79.197.203 (4) - 204.79.197.200 (4) - 72.21.91.29 (4) - 54.245.190.182 (4) - 52.41.2.153 (4) - 52.10.88.168 (4) - 13.152.184.69.138 (2) ... [show more](#)

Analysts can query on a specific MITRE ATT&CK technique to investigate further.

The screenshot shows the RSA Investigate interface with the following details:

- Navigation:** Respond, Investigate (selected), Monitor, Configure, Admin
- Menu:** NAVIGATE, EVENTS, HOSTS, FILES, ENTITIES, MALWARE ANALYSIS
- Query:** adminserver - Broker
- Time Range:** Last 24 Hours
- Filters:** attack.technique = 'hidden files and dir...' | attack.tactic = 'privilege escalation'
- Results:**
 - ATT&CK Technique** [attack:technique] (5 values)
 - hidden files and directories (9) - process injection (9) - spearphishing attachment (3) - scripting (2) - powershell (1)
 - ATT&CK Tactic** [attack:tactic] (5 values)
 - defense evasion (9) - privilege escalation (9) - persistence (7) - execution (3) - initial access (3)
 - Behaviors of Compromise** [boc] (5 values)
 - unsigned creates remote thread (9) - unsigned creates remote thread and file hidden (7) - office application injects remote process (3) - scripting engine injects remote process (2) - powershell injects remote process (1)
 - File Analysis** [analysis:file] (1 value)
 - in hidden directory (9)
 - All IPv4 Keys** [ip:all] (2 values)
 - 192.168.134.138 (9) - 192.168.134.51 (9)
 - Filename Source** [filename:src] (8 values)
 - injector64.exe (2) - cscript.exe (1) - dtf.exe (1) - excel.exe (1) - powerpnt.exe (1) - powershell.exe (1) - winword.exe (1) - wscript.exe (1)
 - Filename Destination** [filename:dst] (2 values)
 - winlogon.exe (7) - explorer.exe (2)

You can also view a query on MITRE ATT&CK keys in Investigation.

The screenshot displays the RSA NetWitness interface. At the top, the 'EVENTS' tab is active. A search query is entered in the search bar: `attack.technique = hidden files and directories AND attack.tactic = privilege escalation AND attack.technique = hidden files and directories`. Below the search bar, a list of 9 events is shown. The first event is selected, showing details for a 'PROCESS EVENT' on 11/05/2019 08:15:37 pm. The event details include: `Injector64.exe PERFORMED createRemoteThread Explorer.EXE`. The metadata fields for this event are: `ANALYSIS FILE: \\hidden\directory`, `ANALYSIS ALL: \\hidden\directory`, `FEED NAME: investigation`, `RISK: low`, `ATTACK TACTIC: defense evasion`, `ATTACK TECHNIQUE: hidden files and directories`, `IOC: unsigned creates remote thread`, `FEED NAME: investigation`, `RISK: low`, `ATTACK TACTIC: privilege escalation`, `ATTACK TACTIC: defense evasion`, `ATTACK TECHNIQUE: process injection`, `ID: endpoint\high\id`, and `RID: 36391`.

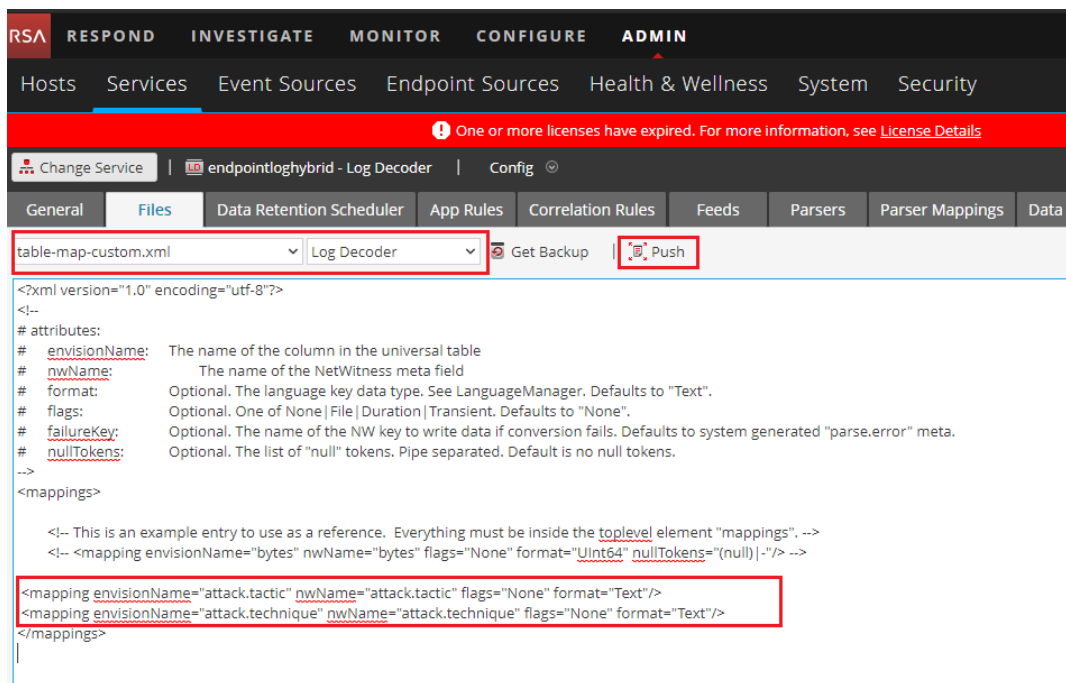
Configure RSA NetWitness for Mitre ATT&CK™ Metadata

In RSA NetWitness 11.4 and above, all MITRE ATT&CK™ metadata is generated out-of-the-box and does not require any customization.

For RSA NetWitness Platform 11.3 and lower, follow these steps to generate the metadata:

1. Add the custom keys, **ATT&CK Tactic** and **ATT&CK Technique**, to the **table-map-custom.xml** file on the Decoder.
 - a. In the **NetWitness** menu, select **ADMIN > Services**.
 - b. In the **Services** grid, select a Log Decoder.
 - c. From the **Actions** menu, select **View > Config**, then select the **Files** tab in the Services Config view.
 - d. Select **table-map-custom.xml** from the drop-down list, and in the `<mappings>` section of the file, add the following new mappings:

```
<mapping envisionName="attack.tactic" nwName="attack.tactic" flags="None"
format="Text"/>
<mapping envisionName="attack.technique" nwName="attack.technique"
flags="None" format="Text"/>
```



- e. Click **Apply** and push changes to other Log Decoders as desired.
2. Add the custom keys, **ATT&CK Tactic** and **ATT&CK Technique**, to the Concentrator custom index file.
 - a. In the **NetWitness** menu, select **ADMIN > Services**.
 - b. In the **Services** grid, select the Concentrator and in the toolbar, select **View > Config**, then select the **Files** tab.

The Device Config view is displayed with the **Concentrator Files** tab open.

- c. Select **index-concentrator-custom.xml** from the drop-down list, and add the new keys as shown below and then click **Apply**.

```
<key description="ATT&CK Tactic" name="attack.tactic" format="Text"
level="IndexValues" valueMax="10000"/>
```

```
<key description="ATT&CK Technique" name="attack.technique"
format="Text" level="IndexValues" valueMax="10000"/>
```

General | **Files** | Data Retention Scheduler | Correlation Rules | Appliance Service Configuration

index-concentrator-custom.xml | Concentrator | Get Backup | **Push**

destination = specifies the key name of the transformed meta value to create

Decoder examples - Normally you do not need to edit index files on the Decoder, unless you want to add aliases or have data privacy requirements. Parsers and feeds declare their meta keys internally and those keys are automatically added to the language. Also, you should *never* set the index level to IndexKeys or IndexValues on a Decoder if you have a Concentrator/Archiver aggregating from it. The index partition size is too small to support any indexing beyond the default "time" meta.

Data privacy
`<key description="existing meta key" format="Text" level="IndexNone" name="existing" protected="true">
 <transform destination="existing.hash"/>
</key>`

Concentrator/Archiver examples - Any new meta keys that should be indexed must be added to this file.

Adding new meta key for custom parser at the index key level
`<key description="my new parser meta key" format="Text" level="IndexKeys" name="mynewparserkey"/>`

Data privacy
`<key description="existing meta key" format="Text" level="IndexValues" name="existing" protected="true">
 <transform destination="existing.hash"/>
</key>
<key description="existing meta key hash" format="Text" level="IndexValues" name="existing.hash" token="true"/>`

Broker derives its language from all the devices it aggregates from. There is simply no need to edit a broker's custom language file.
 -->

`<!-- *** Please insert your custom keys or modifications below this line *** -->`

`<key description="ATT&CK Tactic" name="attack.tactic" format="Text" level="IndexValues" valueMax="10000"/>
<key description="ATT&CK Technique" name="attack.technique" format="Text" level="IndexValues" valueMax="10000"/>`

`</language>`

- d. Push changes to other Concentrators as desired.
 - e. For changes to take effect immediately, restart all concentrators onto which changes were pushed.
3. Get the latest Investigation Feed from Live and deploy to the desired log decoders or decoders.

RSA Respond Investigate Monitor **Configure** Admin

LIVE CONTENT | SUBSCRIPTIONS | INCIDENT RULES | INCIDENT NOTIFICATIONS | ESA RULES | CUSTOM FEEDS | LOG PARSER RULES

Search Criteria
 Keywords:
 Category: [FEATURED] [THREAT] [IDENTITY] [ASSURANCE] [OPERATIONS] [SPECTRUM] [MALWARE ANALYSIS]

Matching Resources
 Show Results | Details | **Deploy** | Subscribe | Package

Subscribed	Name	Created	Updated	Type
<input checked="" type="checkbox"/>	Investigation	2016-11-03 5:41 PM	2019-09-27 6:16 PM	Feed
<input type="checkbox"/>	RSA FirstWatch SSL Blacklist	2017-09-12 9:02 PM	2017-09-22 10:15 PM	Feed

The screenshot shows the RSA web interface with the 'Configure' tab selected. The navigation bar includes 'Respond', 'Investigate', 'Monitor', 'Configure', and 'Admin'. Below the navigation bar, there are links for 'LIVE CONTENT', 'SUBSCRIPTIONS', 'INCIDENT RULES', 'INCIDENT NOTIFICATIONS', 'ESA RULES', 'CUSTOM FEEDS', and 'LOG PARSER RULES'. A secondary navigation bar contains 'Download', 'Subscribe', 'Deploy' (highlighted with a red box), and 'Service Locator'.

The main content area displays the 'Investigation' feed configuration. It includes a table with the following details:

type	Feed
created	2016-11-03 5:41 PM
updated	2019-09-27 6:16 PM
description	The Investigation feed generates metadata based upon the Investigation Model in order to assist an analyst with threat hunting and content generation. This is useful for front line analysts because it minimizes the time dedicated to mining logs or sessions in support of their findings. In order to trigger the feed, a match to an application rule or piece of Lua parser logic is required.

Below the table, there are sections for 'REFERENCES', 'DEPENDENCIES', 'CONFLICTS', and 'INDEXED META KEYS'.

REFERENCES
 More detail can be found on RSA Link at <https://community.rsa.com/docs/DOC-62303>.

DEPENDENCIES
 Parsers
 * FeedParser

CONFLICTS
 Feeds
 * Alert IDs Info
 * Alert IDs Suspicious
 * Alert IDs Warning
 * Hunting

INDEXED META KEYS

TLD Lua Parser Options File

Caution: RSA strongly suggests that you **do not subscribe** to the options file. Subsequent downloads of this file will overwrite all changes that you have made to the file.

Note the following:

- If you deploy the options file, it can be found in the same directory as parsers:
`/etc/netwitness/ng/parsers/`.
- The parser is not dependent upon the options file. The parser will load and run even in the absence of the options file. The options file is only required if you need to change the default settings.
- If you do not have an options file (or if your options file is invalid), the parser uses the default settings.

Note: The parser will never use both the defaults and customized options. If the options file exists and its contents can be loaded, then the defaults will not be used at all.

The `TLD_lua_options` file contains the following options for controlling the parser:

- **deduplicate**
- **localDomains**

To change an option from **false** to **true**, edit the line inside the corresponding function, from

```
return false
to
return true
```

And similarly to go from **true** to **false**.

Note Modifying any of these options requires a service restart to take effect; a simple parser reload is insufficient.

deduplicate

Default value: **true**

By default **tld**, **cctld**, and **sld**, meta are only registered once per session, regardless of the number of times seen.

Disabling this option (setting to **false**) causes the parser to register all tld, cctld, and sld meta even if they have already been registered previously for that session.

localDomains

List of domain suffixes or TLDs specific to the internal network. Multiple values must be comma-separated, for example:

```
"local,here,example"
```

Default value: **local**

Traffic Flow Lua Parser

Introduction

The Decoder identifies the host which initiated a session as **ip.src**, and the responding host as **ip.dst**. However, there is no indication which hosts are internal to your network and which are external to your network. So, there is no indication whether a session was initiated by an internal host to an external host, whether a session was initiated by a external host, or whether a session was between two internal hosts. Nor, for internal hosts, is there any indication where on your network an internal host resides. The traffic flow parser provides this directionality information.

The netblock name (**netname** meta) provides the context of where on your network a host resides. Directionality (**direction** meta) provides the context of whether a session was initiated from an internal host to an external host (outbound), from an external host to an internal host (inbound), or was between two internal hosts (lateral). The following table shows how the network traffic flow direction is determined according to source and destination IP addresses.

Source IP	Destination IP	Traffic Flow Direction
Internal	Internal	Lateral
	Named External	Outbound
	Other	
Named External	Internal	Inbound
	Named External	<none>
	Other	
Other	Internal	Inbound
	Named External	<none>
	Other	

By default, the Traffic Flow Lua parser is not deployed on a Log Decoder for Network Decoder. It should be deployed from Live to enable directionality support. You download the parser from Live to deploy to a Network Decoder, in the same manner as you download and deploy all of the RSA parsers. In addition to the parser, there is an Options file. You only need the Options file if the default settings are not sufficient for your use case. At this time, only manual deployment to a Log Decoder and Network Decoder is supported. The screenshot below shows the Investigation view of these two pieces of meta being populated.

The screenshot shows a header bar with the date '2016 05 26' and time '11:12:00 (+00:00)' on the left, and 'All Data' on the right. Below the header, there are two expandable sections:

- direction** (3 values) with a search icon. It lists: lateral (482) - outbound (53) - inbound (2).
- netname** (7 values) with a search icon. It lists: private src (534) - private dst (401) - multicast dst (58) - other dst (53) - broadcast dst (25) - other src (2) - broadcast src (1).

Setup

The Lua parser and Options file is supported on both the Network Decoder and Log Decoder, but will only deploy through Live to the Network Decoder. Deployment will silently fail for the Log Decoder; if you want the Traffic Flow parser installed on the Log Decoder, you must deploy manually.

The parser and Options file are separate packages in Live. This means that any options file customizations you make do not get overwritten if you update the parser itself. Thus, we do not recommend subscribing to the Options file in Live.

If you need to edit any options, you should deploy the Options file at the same time you deploy the parser itself. Make sure that you reload the parser after you edit the Options file. Make a note that if the default settings cover your network to your satisfaction, then you do not need the Options file.

Deploy to Network Decoders

To deploy the Traffic Flow parser to a Network Decoder:

1. In the NetWitness Suite UI, select **Live > Search**.
2. In the Search Criteria section, type "traffic flow" for **Keywords**.

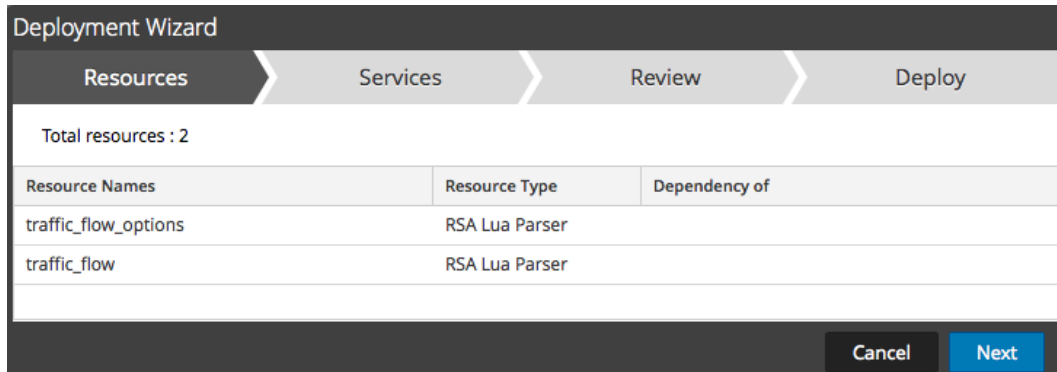
The screenshot shows the 'Search Criteria' section on the left with 'traffic flow' entered in the 'Keywords' field and 'RSA Lua Parser' selected in the 'Resource Types' dropdown. On the right, the 'Matching Resources' section shows a table with two entries:

Subscribed	Name	Created	Updated	Type
<input type="checkbox"/>	traffic_flow_options	2016-06-07 8:41 PM	2016-06-07 8:41 PM	RSA Lua Parser
<input type="checkbox"/>	traffic_flow	2016-06-07 8:41 PM	2016-06-07 8:41 PM	RSA Lua Parser

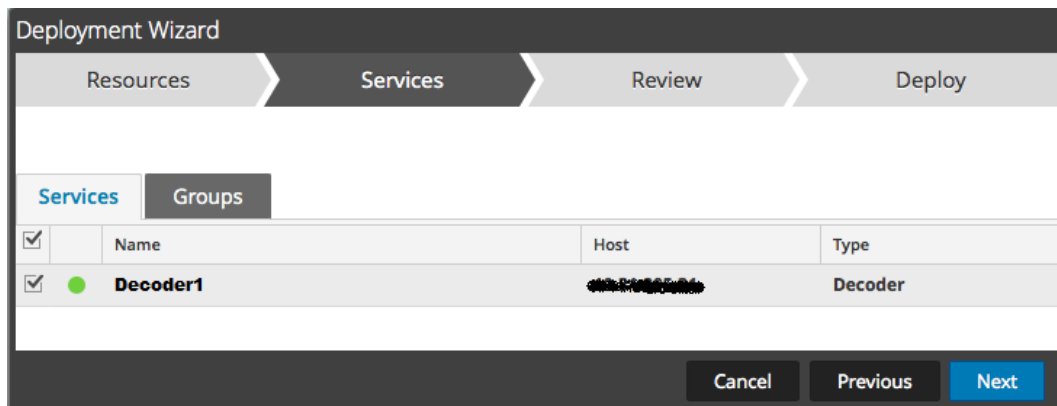
3. In the Matching Resources pane, select both files returned: **traffic_flow_options** and **traffic_flow**.
4. Click **Deploy**.

This launches the **Deployment Wizard**.

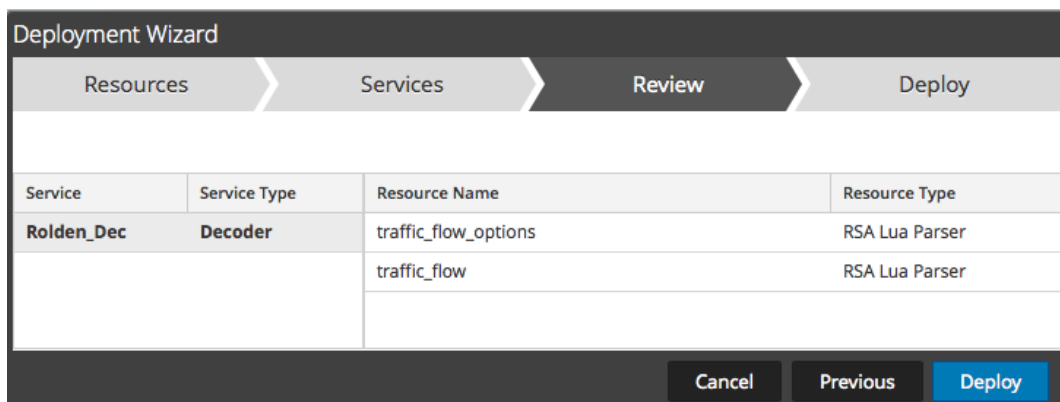
5. Use the Deployment Wizard to deploy to a Network Decoder.
 - a. The two files, **traffic_flow_options** and **traffic_flow**, are shown in the **Resources** screen. Click **Next** to proceed.



- b. In the **Services** screen, select a decoder, and click **Next**.

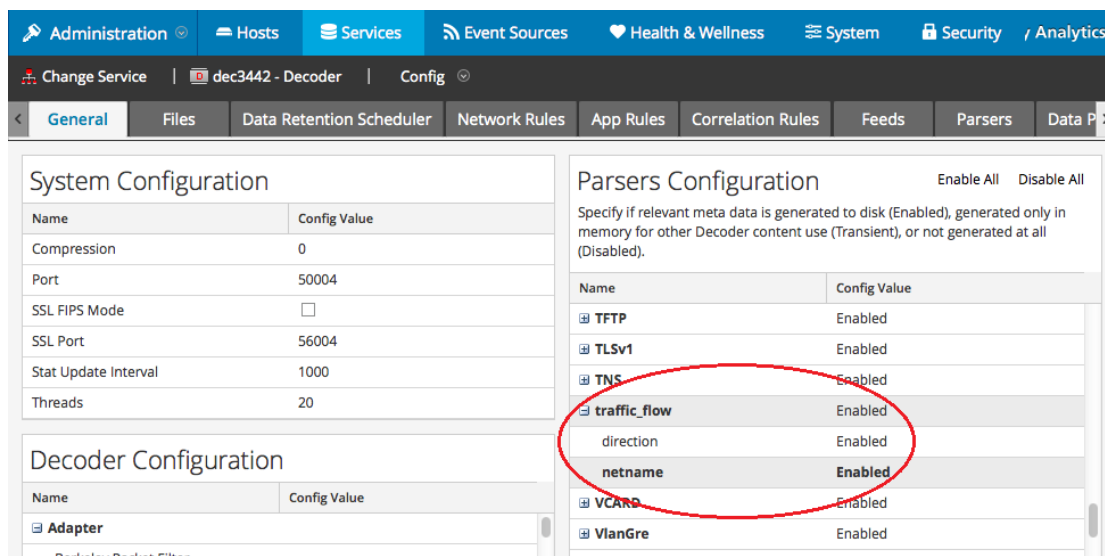


- c. In the Review screen, click **Deploy**.



d. Once the two files deploy, click **Close**.

You can navigate to the **General** tab in the Decoder and view the parser in the Parsers Configuration pane:



Reload Parsers:

Options file is used to change the Traffic Flow Lua parser's option. When you change the Traffic Flow Lua parser's option using the Options file, you must reload the parsers for the changes to take effect. Use the **Explore** view or **NwConsole** to reload the parsers.

To reload the parsers using Explore view:

1. Go to **Admin > Services**. Select the decoder.

2. Click , and in the dropdown select **View > Explore**.

3. Click



. Right Click the **Parsers** in the dropdown.

4. Select **Properties**. Click the dropdown box under **Properties for ... /decoder/parsers**.
5. Select **Reload** and click **Send**.

The parsers have been reloaded message is displayed under **Response Output**.

To reload the parsers using NwConsole:

1. Log in to the decoder using **NwConsole**.
2. Run the command **/decoder/parsers reload**.

Deploy to Log Decoders (For versions Prior to 11.0)

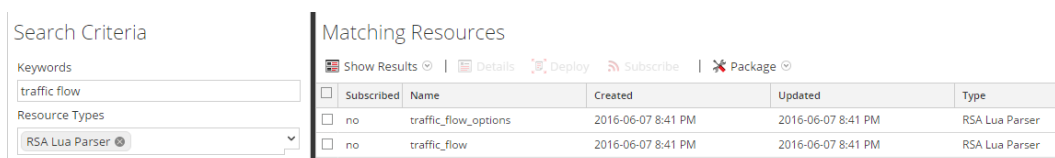
For RSA NetWitness Platform versions 11.0 and later, you can deploy to log decoders using Live, in a similar manner as described in [Deploy to Network Decoders](#) above.

For earlier versions of RSA NetWitness, perform the following steps to install the Traffic Flow Lua parser on Log Decoders.

Warning The **direction** meta is a string; it only contains the last value written to it. If a log parser writes direction meta and the Lua parser is deployed to a log decoder, then that existing direction meta may be overwritten by the parser.


To deploy the parser manually on versions earlier than RSA NetWitness Platform 11.0:

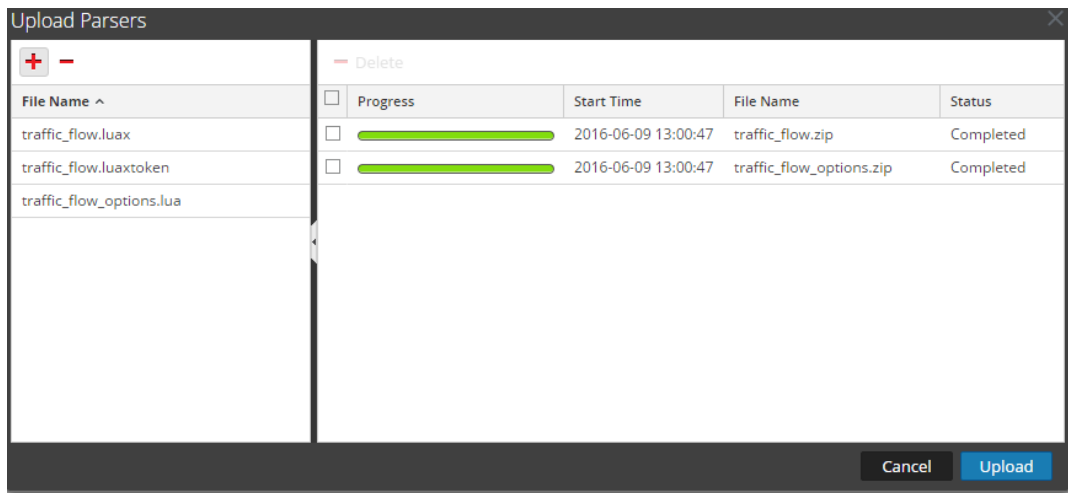
1. Download the Traffic Flow Parser and Traffic Flow Options File from Live.
 - a. In the NetWitness Suite UI, select **Live > Search**.
 - b. In the Search Criteria section, type "traffic flow" for **Keywords**.



- c. In the Matching Resources pane, select both files returned: **traffic_flow_options** and **traffic_flow**.
 - d. Select Create from the **Package** menu.

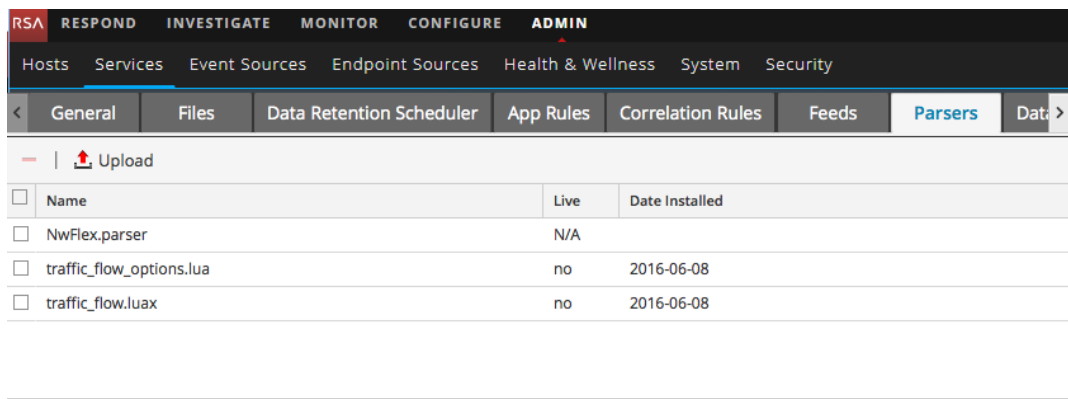
A resource bundle ZIP file is downloaded to your computer.
2. Unzip the file that you downloaded. In the download path, there are two ZIP files, **traffic_flow_options.zip** and **traffic_flow.zip**.

3. Unzip these files: you will see **traffic_flow_options.lua**, **traffic_flow.luax** and **traffic_flow.luaxtoken**.
4. Upload these three files to your Log Decoder.
 - a. In the NetWitness Suite UI, select **Administration > Services** and select a Log Decoder.
 - b. Select **Parsers > Upload**.
 - c. In the Upload Parsers dialog box, click , and navigate to where you saved the parser files on your computer.
 - d. Select all three files: **traffic_flow_options.lua**, **traffic_flow.luax** and **traffic_flow.luaxtoken**.



- e. Click **Upload**.

The files get uploaded, and you can view them in the file list.



5. Restart the Log Decoder service.

Update XML Files

Note: This section only applies to versions prior to 10.6.2. For all newer versions, these entries are delivered automatically.

Add Entries to the Index Concentrator File

You need to add entries to the **index-concentrator-custom.xml** on the Concentrators you would like to index to the meta.

1. Depending on your version:
 - For NetWitness 11.x: In the **NetWitness** menu, select **ADMIN > Services**.
 - For Security Analytics 10.x: In the **Security Analytics** menu, select **Administration > Services**.
2. Select a Concentrator.
3. Select **View > Config** from the Actions menu.
4. Select the **Files** tab, then select the **index-concentrator-custom.xml** file.
5. Add the following lines:

```
<key description="Network Name" level="IndexValues" name="netname"
format="Text" valueMax="10000"/>
<key description="Traffic Flow Direction" level="IndexValues" name="direction"
format="Text" valueMax="10000"/>
```

Add Entries to the Table Map File (Log Decoders Only)

You need to add entries to the **table-map-custom.xml** file on the Log Decoder.

1. Depending on your version:
 - For NetWitness 11.x: In the **NetWitness** menu, select **ADMIN > Services**.
 - For Security Analytics 10.x: In the **Security Analytics** menu, select **Administration > Services**.
2. Select a Log Decoder.
3. Select **View > Config** from the Actions menu.
4. Select the **Files** tab, then select the **table-map-custom.xml** file.
5. Add the following lines:

```
<mapping envisionName="netname" nwName="netname" flags="None"/>
<mapping envisionName="direction" nwName="direction" flags="None"/>
```


Update ESA Configuration

To be able to create ESA rules against the **netname** meta, you need to make netname an array.

To make netname an array:

1. Log onto RSA NetWitness Platform as an administrator.
2. Depending on your version:
 - For NetWitness 11.x: In the **NetWitness** menu, select **ADMIN > Services**.
 - For Security Analytics 10.x: In the **Security Analytics** menu, select **Administration > Services**.



3. Select the ESA service, then **View > Explore**.
4. From the left pane, select **Workflow > Source > nextgen**.
5. Add the meta key name 'netname' to the **ArrayFieldNames**.

Tuning

The following sections describe the parser default values, the options file, and present examples.

Parser Defaults

The default settings for the parser are as follows (these values are in the **internal** table, which replaces the **definitions** table from the earlier version of the options file):

- ["0/8"] = "broadcast"
- ["10/8"] = "private"
- ["127/8"] = "loopback"
- ["169.254/16"] = "link-local"
- ["172.16/12"] = "private"
- ["192.168/16"] = "private"
- ["224/4"] = "multicast"
- ["240/4"] = "reserved"
- ["255.255.255.255/32"] = "broadcast"

If these settings are sufficient, then you do not need to deploy the Options file.

Options File

The Options file contains extensive comments, as well as all the defined default values. Note the following:

- Local subnets must be defined in "internal"
- Named external subnets must be defined in "external". They will result in netname meta, but be considered as "external" for purposes of directionality.
- If a subnet is not listed, it results in **other**, for example , **netname: other src**.
- Aggregation is recommended wherever possible.

For IPv4, note the following:

- Use only CIDR notation. Specifically, don't use a netmask like "255.255.255.0"
- Both shorthand and normal CIDR are valid.
- If netmask is omitted, /32 is assumed.

For IPv6, note the following:

- Use only prefix notation.
- Both compressed and expanded addresses are supported.
- If prefix is omitted, /64 is assumed (not /128).
- Only IPv6 prefixes aligned with nibble (4-bit) boundaries are supported (/128, /124, /120, ... /12, /8, /4). Subnets specified using non-aligned prefixes will not be used.

If you need to define subnets with prefixes that are not nibble-aligned, define multiple smaller prefixes. For example, instead of

```
["2000:a:bc:dee::/63"] = "campus",
```

Use the following:

```
["2000:a:bc:dee::/64"] = "campus",
```

```
["2000:a:bc:def::/64"] = "campus",
```

Caution: RSA strongly suggests that you **do not subscribe** to the options file. Subsequent downloads of this file will overwrite all changes that you have made to the file.

Note the following:

- If you deploy the options file, it can be found in the same directory as parsers:
`/etc/netwitness/ng/parsers/.`

- The parser is not dependent upon the options file. The parser will load and run even in the absence of the options file. The options file is only required if you need to change the default settings.
- If you do not have an options file (or if your options file is invalid), the parser uses the default settings.

Note: The parser will never use both the defaults and customized options. If the options file exists and its contents can be loaded, then the defaults will not be used at all.

Editing the Options File

On the Decoder

To change any of the options for the parser on the Decoder, you must edit the options file itself. You do this from either:

- A shell on the Decoder, or
- In the NetWitness Suite UI: go to the Decoder > Config > Files.

On the Log Decoder

To change any of the options for the parser on the Log Decoder, you must edit the options file itself. You do this from either:

- A shell on the Log Decoder, or
- In the NetWitness Suite UI: go to the Log Decoder > Config > Files.

IMPORTANT: Make sure that you reload the parser after you edit the options file.

Note: Changes take effect immediately. (This is not true for other parsers.)

Option File Details

Starting with version **2017.03.08.1** of the `traffic_flow_options` file, the file contains two tables:

- **internal.** This table replaces the previous **definitions** table. Subnets listed here receive netname meta and are considered internal for directionality.
- **external.** Subnets listed here receive netname meta and are considered external for directionality.

Note: In all prior versions to 2017.03.08.1, there is only one definitions table.

Addresses matching neither table receive netname "other," and are considered external for directionality.

For backwards compatibility, if the parser finds a **definitions** table rather than an **internal** table, then **definitions** is used as **internal**. If the parser finds both, then it uses **internal**.

The tables **internal** and **external** for 'traffic_flow' are Lua tables that contain a list of netblocks in CIDR notation. The syntax of each entry is:

```
["ip/prefix"] = "name",
```

For example:

```
["192.168.0.0/16"] = "private",
```

Shorthand notation is valid. For example, the following entry is equivalent to the previous example:

```
["192.168/16"] = "private",
```

Caution: All aspects of the syntax are crucial, including the trailing comma. If the options file is edited so as to become invalid, the parser will use default values.

Be sure to account for all of your environment's internal networks (but only these).

If a specific entry in the lua table isn't valid CIDR notation, it will be ignored. However, it will not affect the rest of the table (so long as the table syntax is valid) and an error will be logged.

For example:

```
["10.1.1.0/a"] = "dmz",
```

This results in the following logged error message:

```
traffic_flow: invalid cidr '10.1.1.0/a'
```

Matching Rules

For matching, the most specific prefix for an IP "wins". For example, assume the options file has both of the following entries:

```
["192.168.0.0/16"] = "private",
["192.168.1.0/24"] = "dmz",
```

The IP **192.168.1.1** will match "dmz", and the IP **192.168.2.1** will match "private".

The following keys will be matched against the networks listed in the table:

- ip.src
- ip.dst
- alias.ip
- ip.addr
- orig_ip

Examples

For each value of each of the above keys, meta is registered with the key "netname" as follows:

- For ip.src: *<name from entry>* src
For example: `private src`
- For ip.dst: *<name from entry>* dst
For example: `netname: private dst`
- For the other keys: *<name from entry>* misc
For example: `netname: private misc`

If a value does not match an entry in the table, then the name "other" is used instead:

- Example for ip.src: `other src`
- Example for ip.dst: `netname: other dst`
- Example for the other keys: `netname: other misc`

Further, for each src and dst pair meta "direction" is registered:

- If ip.src is listed and ip.dst is "other":
`direction: outbound`
- If ip.src is "other" and ip.dst is "listed":
`direction: inbound`
- Both ip.src and ip.dst are listed, or neither are listed:
`direction: lateral`

Here is an example of what the options file looks like for versions 2017.03.08.1, and newer, where the file contains both the internal and external network definitions:

Note: Some application rules leverage “direction = outbound” to help minimize false positives and reduce noise. By listing web proxies as external, this allows these app rules to function as expected if you have NetWitness visibility on the ‘inside’ of your web proxy. If NetWitness has visibility on the ‘outside’ of your proxy, there is no need to set your web proxies as ‘external’, and the app rules will function as intended.

```
function internal()  
--=[ INTERNAL NETWORKS  
    For proper direction meta:  
        (a) add all internal subnets  
        (b) DO NOT add any external subnets here  
--]=]  
return {  
    ["0/8"] = "broadcast",  
    ["10/8"] = "private",  
}
```

```

["127/8"] = "loopback",
["169.254/16"] = "link-local",
["172.16/12"] = "private",
["192.168/16"] = "private",
["192.168.2.101/32"] = "HOST vineyard",
["224/4"] = "multicast",
["240/4"] = "reserved",
["255.255.255.255/32"] = "broadcast",
}
end

function external()
--=[ NAMED EXTERNAL NETWORKS
    For proper direction meta:
        (a) DO NOT add any internal subnets here
        (b) add any desired external subnets
--]=]
return {
    ["1.2.3.0/24"] = "partner network vpn",
    ["104/8"] = "TESTNET-1",
    ["151.101/16"] = "TESTNET-2",
    ["172.16.0.0/24"] = "Web proxies",
    ["65.52/16"] = "DMZ",
    ["52.84.126.141/32"] = "KEEP_AN_EYE_ON_THIS_SYSTEM",
}

```

The following is an example of the definitions table for prior versions of the options file (that contained only one table):

```

function definitions()
return {
    ["0/8"] = "broadcast",
    ["10/8"] = "privatel0",
    ["10.10.72.0/21"] = "topeka-lab01",
    ["10.10.80.0/21"] = "topeka-voip",
    ["10.10.88.0/21"] = "topeka-voip",
    ["10.10.96.0/21"] = "topeka-vpn",
    ["10.10.140.0/19"] = "campus-works",
    ["127/8"] = "loopback",
    ["169.254/16"] = "link-local",
    ["172.16/12"] = "privatel172",
    ["192.168/16"] = "privatel192",
    ["192.168.2.253"] = "***<<SourceCode_Server>>***",
    ["192.168.121.0/24"] = "hydrotestlab",
    ["192.168.150.101"] = "corporate-svr01",
    ["192.168.150.102"] = "corporate-svr02",
    ["192.168.150.103"] = "corporate-svr03",
    ["192.168.150.0/20"] = "corporate-netp150",
    ["224/4"] = "multicast",
    ["240/4"] = "reserved",
    ["255.255.255.255/32"] = "broadcast",
    ["201.101.141.0/24"] = "ACMElabB1F01R101",
    ["201.101.142.0/24"] = "ACMElabB1F01R102",
    ["201.101.143.0/24"] = "ACMElabB1F01R103",
}

```

```
["201.101.144.0/24"] = "ACMElabB1F01R104",
["208.43.253.0/24"] = "brooklyn-b2f11012",
["217.43.253.0/24"] = "brooklyn-b2f21012",
["222.43.253.0/19"] = "brooklyn-b2f31012",
["222.44.253.0/24"] = "brooklyn-b01SF",
["228.100.17.0/26"] = "brooklyn-DMZ01",
["228.100.18.0/27"] = "brooklyn-DMZ02",
["88.56.104.0/20"] = "remoteOffice-R201",
["147.22.56.0/21"] = "Branch-Services",
}
```

Key Mappings and Defaults

Keys used for source (*netname_src*) and the determination of directionality:

- ip.src
- ipv6.src

Keys used for destination (*netname_dst*) and the determination of directionality:

- ip.dst
- ipv6.dst

Keys used only for miscellaneous (*netname_misc*) not the determination of directionality:

- alias.ip
- ip.orig
- alias.ipv6
- ipv6.orig
- ip.addr
- ipv6.addr

Full default values:

INTERNAL

- ["0/8"] = "broadcast",
- ["10/8"] = "private",
- ["127/8"] = "loopback",
- ["169.254/16"] = "link-local",
- ["172.16/12"] = "private",

- ["192.168/16"] = "private",
- ["224/4"] = "multicast",
- ["240/4"] = "reserved",
- ["255.255.255.255/32"] = "broadcast",
- ["fc00::/8"] = "unique-local",
- ["fd00::/8"] = "unique-local",
- ["fe80::/12"] = "link-local",
- ["fe90::/12"] = "link-local",
- ["fea0::/12"] = "link-local",
- ["feb0::/12"] = "link-local",
- ["ff00::/8"] = "multicast",

EXTERNAL

- ["ff0e::/16"] = "multicast global",
- ["ff1e::/16"] = "multicast global",

Configure Windows Collection

Overview

This topic provides details about configuring Windows collection so that NetWitness Suite can collect logs from Microsoft Windows machines. In this document, the word "Collector" refers to either the NetWitness Suite Log Collector or the NetWitness Suite Virtual Log Collector. The word "Channel" refers to a Windows Event Log, for example, a Security Application, System, Forwarded Event, or DNS.

Windows Eventing Collection is the collection of events from Windows systems using the Windows Remote Management (WinRM) protocol, and is a Microsoft implementation of the WS-Management protocol, which runs as a service on most Windows desktops and servers. This service uses HTTP or HTTPS as its transport mechanism. The request / responses are wrapped in a SOAP envelope (a simple XML wrapper) to give them structure.

The WinRM service is not set up to run by default on all Windows operating systems, nor is it configured to listen for requests by default, which means that in order to use WinRM, you must configure it on the systems you are using. The *Microsoft WinRM Configuration* guide describes how to configure the WinRM service and its sub layers, either manually or by using group policies, to allow the Collector to collect Windows event logs. The *Test and Troubleshoot Microsoft WinRM* guide provides detailed information about resolving WinRM configuration issues. These documents are available as a PDF Portfolio on [RSA Link](#) here: [Microsoft WinRM Configuration and Troubleshooting](#).

In its simplest form, the WinRM protocol is used by the Collector to send a **subscribe** request that contains filters, such as channels (event log names) and exclusions (or inclusions), for certain Windows event types. Bookmarks, which are also included in subscribe requests, are numbers that correspond to record IDs in Windows event logs. NetWitness Suite stores these numbers so that if the Collector is restarted, it can restart where it left off from where it was collecting from previously. They are actual event record IDs which you can view in the corresponding Windows event log by highlighting an event and clicking **Properties**. When a **subscribe** request is successful, it is followed by successions of **pull** commands to retrieve events from a target system, and finally, an **unsubscribe** request is sent to end the poll cycle. To authenticate to the Windows system, NetWitness Suite allows you to select Basic or Negotiate (Kerberos) authentication types in the event category section of an event source to provide credentials to the target system while establishing a connection.

When the event category of an event source is configured for the Negotiate authentication type (Kerberos is the default protocol), the Collector requires access to a domain controller via port 88 UDP. The Collector needs to retrieve a TGT for the log collection user account, and an ST for each of the Windows event sources being collected from in order to provide credentials for access to the WinRM service on each system. If a firewall exists between the Collector and the domain controllers, the firewall must allow at least UDP traffic on port 88 inbound to the domain controllers.

In order to send requests to Windows systems, a listener must be available on each system, that is, logic in the WinRM service that opens a TCP port to listen for incoming requests. In deciding whether to use HTTP or HTTPS as the type of protocol, consider the following information:

- If you use Basic authentication type in the NetWitness Suite event source configuration, using HTTP as the WinRM transport protocol (when creating the listener) is not advisable. Since HTTP is not encrypted, and Basic authentication merely adds Base64-encoded credentials to the header of the request to Windows systems, this is not secure, hence the caution below.
- If you use Negotiate authentication type in the NetWitness Suite event source configuration with HTTP, the credentials are already passed in the form of an encrypted Windows token, so no leakage of credentials will occur. However, the event log payload from the systems is in the clear – it is not encrypted.
- The HTTPS transport with either Basic or Negotiate authentication type encrypts both credentials and payload at the expense of being more difficult to configure across a large number of Windows systems.

Caution: When you use Negotiate as the authentication type, the Windows token that is passed to the target system is encrypted. However, if you use Basic as the authentication type, and you use HTTP, the Log Collection user's user ID and password are sent in the HTTP header with only Base64 encoding, which is not secure. If you use Basic as the authentication type, RSA recommends using HTTPS.

When a collector poll cycle begins, a TCP connection to the WinRM listener port on the target system is established. The default port for this connection is 5985 for HTTP, and 5986 for HTTPS, but these can be overridden by using manual commands to create the listener, or via GPO. If a firewall exists between the Collector and the target systems, you must configure a rule to allow at least inbound TCP connections to those systems on the WinRM ports.

Note: If you are using *quickconfig* and requesting HTTP, the firewall rule is applied automatically. For more information, see [LINK TO QUICKCONFIG](#). When an event category is configured for Basic authentication, you only need a WinRM firewall port rule for each system being collected from. Because no Kerberos connection is used, a connection from the Collector to the domain controller is never created.

Note: The NetWitness Suite Log Collector can be used to collect Windows events from domain controllers, non-domain controller systems in a domain, and workgroups. For domain controllers and non-domain controller systems in a domain, it is advisable to use a domain account. (In fact, you must do this for a domain controller.) A standalone system or a workgroup-only system requires that a local account be used.

There are two areas that complicate WinRM deployment, especially if you are using Windows Group Policy Object (GPO):

- Using HTTPS transport instead of HTTP (see explanation of each above)
- Using a non-Administrative account as the Log Collection user account (which is highly recommended). A non-Administrative account on either domain-based, standalone or workgroup systems must have certain permissions to successfully connect and to read events.

The steps to create these permissions can be performed manually on each target system or by a combination of using GPO and scripting. (Currently, there is no way to enable WMI read rights with GPO, which affects SID enumeration by the Collector, or a reliable way to enable an HTTPS listener via GPO.)

You can use an RSA supplied script to accomplish these tasks. See the *Microsoft WinRM Configuration* guide for details on how to configure an HTTP or HTTPS listener on a system and to set permissions for a non-administrative account to collect from that system. The same script can be pushed as a logon script via GPO to apply the same configuration across a broad number of systems. RSA recommends that you perform a test run of the script on a lab system in a lab to observe what it does, before pushing it out in a large scale manner via GPO. See the *Microsoft WinRM Configuration* guide for a full list of script features.

The following user permissions must be enabled on each system from which events are collected for a non-Administrative Log Collection user account:

- Windows Management Instrumentation (WMI) Remote Enabled permission
- WMI **Read** permission (cannot be enabled with GPO)
- Membership in the built-in Event Log Readers group to physically access the event logs (cannot be enabled with GPO, but for domain systems only, you can add the Log Collection user account to the domain-level Event Log Readers group).

Finally, apart from the listener and non-administrator user complications, because the WinRM service on Windows systems runs with Network Service account privileges by default, the Windows Security Event Log requires an Access Control List (ACL) to allow the Network Service account to read from it. This step can be easily accomplished either manually or with GPO.

In summary, while enabling WinRM in your environment, there are some choices that should be made up front. These are:

- What type of Log Collection user account do I use (administrator vs. non-administrator)?
Non-administrator is highly recommended to limit exposure to administrator credentials. Even though administrator credentials are stored in NetWitness Platform in an encrypted lockbox, RSA still recommends not using an administrator account.
- Should I use HTTP or HTTPS?

- HTTP should not be used with Basic authentication, since it exposes the credentials in the header of the request to the system being collected from.
- When you use Negotiate authentication with HTTP, the credentials (in this case, a Windows Service Ticket) are encrypted, but the event log data payload is not.
- There are two ways to use HTTPS; with, or without mutual authentication. In either case, the payload is encrypted and systems from which event data is being collected must have valid certificates that have at least client and server authentication enabled in the Enhanced Key Usage (EKU) bits. Using full mutual authentication requires the extra step of installing each system's certificate on the Collector using the NetWitness Platform user interface, which provides the means for the Collector to verify the host from which it is collecting the logs. Full mutual authentication requires that the Collector can also reverse look-up the IP address of the system to verify the hostname in the certificate's domain name. If this level of verification is not required, setting up HTTPS is a little simpler, since installing a certificate from each system can be cumbersome.
- Do I enable settings manually or by using GPO?
 - In most production environments, using a GPO is the preferred method of enabling WinRM. This offers good flexibility for the machines that are targeted (for example, IP, subnet, or a machine list from Active Directory). One drawback is that if you are using HTTPS or a non-administrator user, there are steps that cannot be done directly with GPO. These steps must be performed either manually or with a login script, as described in the *Microsoft WinRM Configuration* guide.
 - Using a GPO is a good way to push certificates to systems if HTTPS is required, rather than manually creating one per system (however, this can be done by auto-enrollment).
 - The decision can come down to the numbers of systems involved: If there are only a relatively small number of systems, the manual method may be much faster than testing, getting approval, and finally pushing a GPO out.

The first step is to create the non-Administrative collection user account (if none already exists) as described in the next section.

Create a User Account for RSA NetWitness Platform

RSA recommends that the user account that RSA NetWitness Platform uses to authenticate to the event source has only enough privileges to allow event collection.

If your event source is a part of a Windows domain, you must have your domain administrator create a user account on the domain controller with a password that is sufficiently complicated as per your company policies. If this password is set to expire, remember that collection will stop when this occurs. Hence the password must be maintained (refreshed) outside of NetWitness Platform, and the event sources in NetWitness Platform updated when the password is changed in the future. If the event sources are not part of any domain (for example, standalone or workgroup systems), you must create this user account on each of the individual systems being collected from (event sources).

For standalone systems, create a local non-administrator user account:

1. On the event source, click **Start > Administrative Tools > Server Manager** to open the Server Manager console.
2. Use Server Manager to create a new user account with the following parameters:

Note: You must create one user account for each domain you want to collect from. Ensure that there are no local accounts with the same user name.

- **User name:** Enter a user name for the account.
For example, logcollector.
- **Full name:** Enter a full name for the user account.
- **Description:** Enter a description of the user account.
For example, Account for remote collection of events in RSA NetWitness Log Collector.
- **Password:** Enter a strong password, and select **User cannot change password** and **Password never expires**.

After you create users, you must verify the WinRM listener and the assignment of privileges to the non-Administrative user. This procedure varies depending on whether you are using the GPO or manual mode for configuration. Please follow the steps in the *Microsoft WinRM Configuration* guide.