RSA NETWITNESS® PLATFORM

Test and Troubleshoot Microsoft WinRM

Trademarks

License Agreement

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.

November 2018

Contents

Introduction

The RSA NetWitness Log Collector collects logs from various Windows event log sources using the Windows Remote Management (WinRM) service. The Microsoft WinRM Configuration Guide discusses the methods and protocols. This document describes various methods of end-to-end troubleshooting this method of collection.

Note: An **event source** and a **target** are synonymous with a Windows server, standalone system, or desktop that events are to be collected from. **Channels** equate to Windows event logs, and **Listeners** are ports opened locally by the Windows Remote Management service on the targets to receive requests.

WinRM (Windows Remote Management Service) is a service that was added in Windows Vista, Windows Server 2003 R2 and Windows 2008. It consists of a service (the WinRM component) that exposes various APIs.

The WinRM service can, for example, provide access to the WMI (Windows Management Instrumentation) layer within Windows. WMI provides a wealth of information from the hardware layer up to the UI. The Collector uses WinRM to collect from event channels on target systems. There are typically four event channels of interest while collecting events via WinRM (although other channels can be collected from):

- Security
- Application
- System
- ForwardedEvents

Out of Security, Application, and System event logs, the Security logs contains by far the most useful information. Most of the events in the Application channel are created by applications and cannot be parsed by RSA NetWitness. An exception to this would be, for example, that if the system is an Oracle server, some events collected via the Application channel could be parsed using the ODBC parser. However, it is not uncommon for customers to collect only from the Security logs.

ForwardedEvents is a special channel where a single system can be configured to collect events logs from any event channel on multiple systems in a domain. The RSA NetWitness Log Collector in turn can collect logs originating from the other systems from the single system (which acts as an Event Log Concentrator) using the ForwardedEvents channel.

Note: Windows systems require credentials to access event logs. RSA NetWitness supports two types of authentication, Basic or Negotiate (Kerberos).

Authentication and Encryption

Typically, connections to a Windows target cannot occur without some sort of authentication (unless security has been disabled). Three Authentication types are allowed for access to WinRM :

1. Negotiate (Kerberos or NTLM, Kerberos authentication by default).
2. Digest (not supported by Log Decoder).
3. Basic authentication.

Note: Even if the HTTP transport type is selected, Kerberos and NTLM authentication is still encrypted.

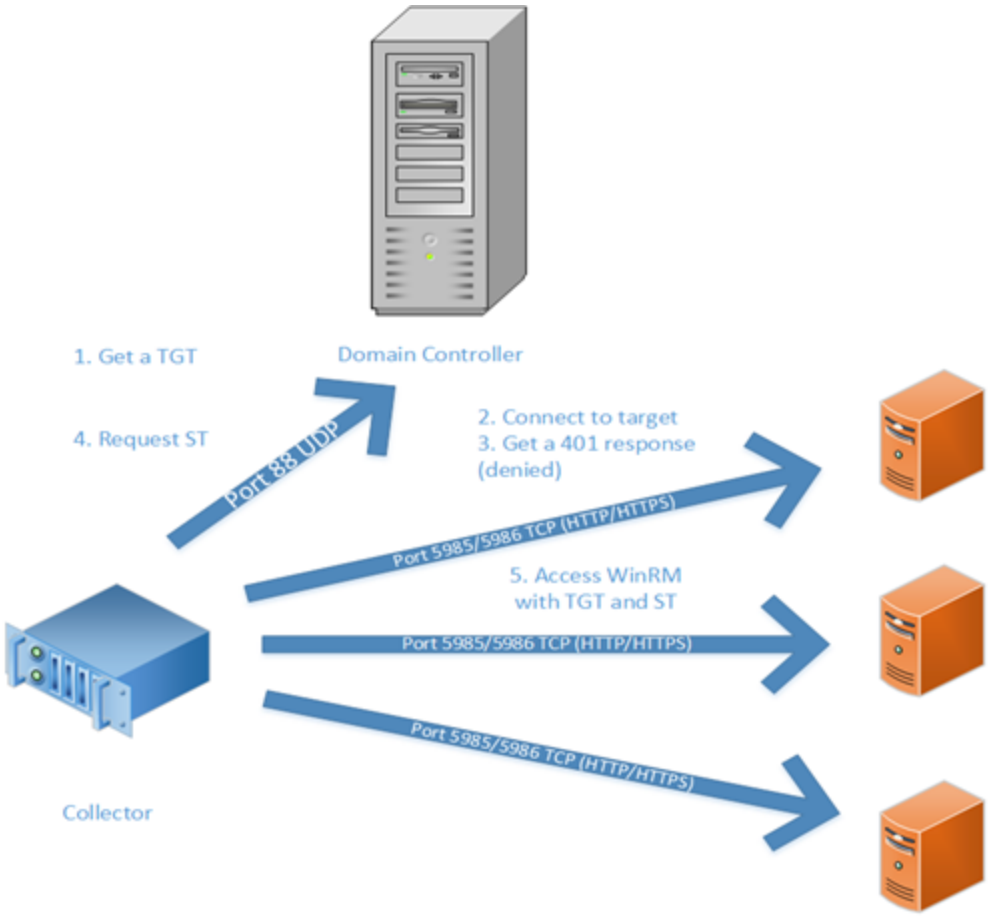
Basic Authentication is typically only used for standalone systems (workstations that are not part of a domain) where a local account MUST be used as opposed to domain credentials. Basic is not recommended, as you would have potentially a different set of credentials per target, which is difficult to manage. If you do use Basic, RSA recommends that you use HTTPS: otherwise, a sniffer could intercept credentials within packets that are transferred between the Collector and the target.

Note: A local account must be used for Basic, not a domain account. If you must use Basic, you cannot use it when collecting from a domain controller. However, for domain member systems, a local account can be used.

Communication

When the event source is configured for Negotiate authentication type, the Collector must first retrieve Windows tokens, that is, TGTs (Ticket Granting Tickets) using each collection user's credentials for the configured domains. Once a TGT is acquired for the domain in which the current Event Source (target system) being collected from is located, an attempt is made to connect to the listener on the target system using the port and protocol configured in the event source. If a listener is available, the attempt will be rejected with a 401 error, since the Collector's initial connection carries no valid tokens. If a 401 error is received, the Collector will try to acquire a Service Ticket (ST) from the domain controller. The Collector will then connect to the listener on the target, again passing the TGT and ST (for that system) encrypted in the HTTP header. If this is accepted, WinRM verifies that the **AllowUnencrypted** flag is set if the listener is HTTP (not HTTPS), and if the flag is NOT set, the request is rejected with a 500 error. A check is then made to verify if the user has remote access enabled, and if not, then the connection will fail with **unexpected HTTP error code 0**. At this stage, if all is good, the Collector issues a **subscribe** command for the channels (event logs) that are configured in the event source (or System, Security and Application if the field is left blank). The final check by the WinRM service is to verify that the user is a member of an Event Log Readers group (if user is non-Admin). If the user is not a member, then system returns a 401 error. If the check is successful, the system returns a positive response to the **subscribe** command and the Collector issues a succession of **pull** commands to retrieve events.

The following diagram shows the flow when using Negotiate. (For Basic only step 5 applies, where the credentials are sent on the request to the **/wsman** URL base64 encoded, instead of the TGT and ST. After that, the flow is the same, that is, if an **HTTP continue** response is received, a **subscribe** command is sent and collection begins):


1. Get a TGT
Domain Controller
4. Request ST
Port 88 UDP
2. Connect to target
3. Get a 401 response
(denied)
Port 5985/5986 TCP (HTTP/HTTPS)
5. Access WinRM
with TGT and ST
Port 5985/5986 TCP (HTTP/HTTPS)
Port 5985/5986 TCP (HTTP/HTTPS)
Collector

Common Issues

Here are the questions to answer while troubleshooting:

- Is this an issue between the Collector and the local domain controller?
- Is it a network issue between the Collector and the target event source?
- Is there a problem with the Windows Remote Management listener on the system being collected from?
- Is there an issue with collection user permissions on the system being collected from?
- Are security events not being collected, while other events are being collected?

Collector/Domain Controller issues

- Getting tickets from the configured domain controller completely fails.
- No ST (Service Ticket) for the host being collected from.
- After an upgrade collection from all Windows event sources fails with 401 errors in **/var/log/messages**
- Collection used to work then one day collection from all Windows event sources fails with 401 errors in **/var/log/messages**

Collector/domain controller issues will always result in 401 errors in **/var/log/messages** or when clicking the **Test** button in the event source.

When using the Negotiate authorization type in an event source, the first step is to figure out just which tickets, if any, were retrieved:

1. If there is no TGT, then the issue is with Kerberos. Follow the methods described in this document to verify the Kerberos configuration, userid, and password. Typically, the issue is either with the Kerberos configuration file on the Collector, **/etc/krb5.conf** (normally, if this file is not hand-edited, and domain controllers are added by using the UI, this is not a common issue), a communications issue with the domain controller (DNS), or finally, time skew errors which are reported using the **kinit** command (and are easy to spot).
2. If there is a TGT for the domain with the failing systems but no ST for those systems, what is the error being returned:
    a. If the error contains “Windows Collection not running,” the listener is not configured on the remote system, or access is blocked to its port.
    b. If it is a 401 error, the issue is most likely a local Kerberos issue. Follow the steps to verify that the configured hostname resolves correctly.

3. If there is both a TGT (token in cache that begins with **krbtgt/**) and an ST (token in cache that begins **HTTP/**) then the issue is typically on the Windows system itself. What is the error code:
   a. 401: Check that collection users are members of an Event Log Readers group (domain-level group when the target is a domain controller or local Event Log Readers group when it is a domain member or standalone system).
   b. If the error contains the string **The cluster resource is not online**, then the security event log ACL has not been modified to include read access for the Network Service account.

Note: Without a listener on a target system being collected from, no attempt will be made by the Collector to acquire an ST for the system!

To troubleshoot Collector and domain controller issues:

1. SSH to the Log Decoder or Remote Log Collector (the system that is configured for the event source of the failing system).
2. Run the following command:
   ```
   export KRB5CCNAME=DIR:/var/netwitness/logcollector/runtime/krb5_ccache_dir
   ```
3. On the Log Decoder or Remote Log Collector where the Collector is located, run:
   ```
   klist -a
   ```
   a. Check for a non-expired TGT (an entry in the **Service Principal** column that begins with **krbtgt/** and ends with the domain being collected from).
   b. Check for a non-expired ST (an entry in the **Service Principal** column that begins with **HTTP/** and ends with the FQDN of the system being collected from) in the Kerberos cache.

The following example shows a typical klist output:

```
Ticket cache: DIR::/var/netwitness/logcollector/runtime/krb5_ccache_dir/tktdawC0c
Default principal: collectoruser@2K8R2-VCLOUD.LOCAL

Valid Starting    Expires           Service Principal
02/03/16 07:41:51  02/03/16 17:41:51 krbtgt/2K8R2-VCLOUD.LOCAL@2K8R2-VCLOUD.LOCAL
renew until 02/10/16 07:41:51
Addresses: (none)
02/03/16 07:43:51  02/03/16 17:41:51 HTTP/2k8r2-dc1.2k8r2-vcloud.local@2K8R2-VCLOUD.LOCAL
renew until 02/10/16 07:41:51
Addresses: (none)
```

Note: There is only one **krbtgt** (TGT) entry in the cache per domain configured, which means that if you configure multiple Event Categories in the Log Collector for the same domain with different collection user accounts, strange results will ensue since a TGT should be unique for that domain. What happens is that a new TGT is requested, which can overwrite previous TGTs.

If the Kerberos cache contains both a valid TGT and ST for the target system, it's very likely that the issue is on the Windows system. The investigation moves to the system being collected from. See the Winrmconfig script section for details on what to do next.

If either of the TGT or ST tickets above have expired or do not exist, the issue is between the Collector and the domain controller and this MUST be fixed first. The highlighted sections of text in the example below are the key parameters that are necessary to get Kerberos to work for the Collector (with only those settings it should work, which means if a ticket cannot be acquired, those are the settings to check).

```
[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[libdefaults]
default_realm = VCLOUD.LOCAL
dns_lookup_realm = false
dns_lookup_kdc = false
ticket_lifetime = 24h
renew_lifetime = 7d
forwardable = true
rdns = false

[realms]
VCLOUD.LOCAL = {
kdc = DC1.VCLOUD.LOCAL
admin_server = DC1.VCLOUD.LOCAL
}

[domain_realm]
.vcloud.local = VCLOUD.LOCAL
vcloud.local = VCLOUD.LOCAL
```

The realms section is very important, specifically the **kdc** entry, which should be the FQDN of the domain controller, and the domain portion which should be in upper case. Note the tag:

```
VCLOUD.LOCAL =
```

Ensure that this is the only the actual domain, and not the FQDN of the domain controller.

Now that **klist** has spotted missing tickets, it becomes a question of WHAT is missing. If there is no TGT, consider the following questions:

- Is there a format issue with the **/etc/krb5.conf** file?
- Are there connectivity issues with the configured **kdc**?

You can use **kinit** to quickly identify if either of these issues are the problem.

Note: The Kerberos cache export command MUST be executed BEFORE using kinit. If kinit is used without running the export command below, it will effect where Kerberos writes tickets on behalf of the Collector and can cause collection to fail for systems that were working.
`export KRB5CCNAME=DIR:/var/netwitness/logcollector/runtime/krb5_ccache_dir`

After running the export command above, run the following command:

```
kinit user@DOMAIN.COM
```

where `user@DOMAIN.COM` is your collection user account. Kerberos tries to authenticate to the domain in the **/etc/krb5.conf** file that matches the domain portion of the account.

The following error messages show reasons that the **kinit** command would fail if there IS actual communication with the domain:

- `kinit: Preauthentication failed while getting initial credentials`
  Usually indicates a password error.
- `kinit: Client not found in Kerberos database while getting initial credentials`
  Domain specified is correct but the username is wrong.
- `kinit: Cannot find KDC for requested realm while getting initial credentials`
  The **/etc/krb5.conf** file has syntax errors, see example below for hints.
- `kinit: KDC reply did not match expectations while getting initial credentials`
  The domain controller is not contactable (possible DNS issue). See below for methods to test connectivity.
- `kinit: clients credentials have been revoked while getting initial credentials`
  The account has been locked out or deleted.
- `kinit: Cannot contact any KDC for realm '`*`YOURDOMAIN`*`' while getting initial credentials`
  The **kdc** was not contactable. See tips below on testing connectivity.

Sometimes the speed at which **kinit** fails is enough to point out that the problem is the format of the **krb5.conf** file, for example, if the failure is very fast. If a communications issue is the problem, it usually takes up to 30 seconds to fail. Opening another **ssh** session to the Collector and running a **tcpdump**, as follows, will show if there are communications issues with the domain controller:

```
tcpdump port 88
```

If no traffic flow appears in the **tcpdump** when the **kinit** is run, the problem is most likely the format of the **/etc/krb5.conf** file.

If there are only packets going to the domain controller, but none are received (just SYN frames), either a firewall is blocking port 88 UDP, or the hostname of the **kdc** in the **/etc/krb5.conf** file does not resolve to the correct IP address.

- Check **/etc/hosts** for any incorrect entries for the **kdc** FQDN (domain controller).
- Check **/etc/resolve.conf** for the correct DNS server.
- Perform an **nslookup** on the FQDN **kdc** and see what it resolves to.
- Ping the FQDN and see if there is a response.
- If the IP address is correct and it does not respond to pings, and **tcpdump** shows only outbound SYN frames, check with the firewall administrative team.

If a TGT is retrieved, that is, it appears in the Kerberos cache, but no ST ever appears, or only appears for certain target systems, it is likely that reverse DNS entries do not exist or are wrong for certain systems.

First, edit the **/etc/krb5.conf** file and add the following parameter to the **libdefaults** section as shown in the **krb5.conf** example above:

```
rdns = false
```

Note: Host names alone should NOT be used in event source address fields for target systems. If an IP address is used in the event source instead of a FQDN, the same thing applies; reverse DNS must be available, or the **rdns = false** setting must be used.

This will fix the common reverse DNS problem with no need to restart anything. To retest the failing event source, just click the **Test** button. If the 401 errors still persist, but a fresh **klist –a** command now shows that an ST for each failing system IS now there, the problem moves from there to the target systems themselves. For information about this, see Winrmconfig script.

Note that the **rdns** setting will fix reverse DNS issues at the expense of a less secure Kerberos connection. If required, you can make the following checks to figure out why the DNS is not working correctly:

- Check **/etc/hosts** for any incorrect entries for the failing target systems.
- Check **/etc/resolve.conf** for the correct DNS server.
- Perform an **nslookup** for the FQDN of the failing target systems. Note the IP address that is returned and then do a **nslookup** for that IP address and see if it matches the FQDN of the system or if it returns anything at all.

At the end of this section, both a TGT and an ST per configured target system should now be available. If not, contact Support and open a case to investigate this issue. With the ST available for a given system, the Collector will attempt a TCP connection to that system on the port specified in the event source. If clicking the **Test** button or watching **/var/log/messages** for the FQDN of that system reveals transport errors, see the following section. If a **401**, **500** or **unexpected HTTP error code(0)** error occurs (in effect any error at this stage that is not a **transport** error) go to Winrmconfig script for further troubleshooting.

Collector/Target tcp connectivity issues (transport errors)

This section describes the typical connectivity issues that can occur between the Collector and target systems, specifically, the inability to create a TCP connection to the target system.

[WindowsCollection] [failure] [DC3.2k8r2-dc1_2k8r2-vcloud_local2] Error subscribing. Transport error code = 6/Could not resolve host.

The error above indicates that there is no DNS entry for the hostname specified in the Event Category. Check **/etc/resolve.conf** for DNS server entries and **/etc/hosts** for hard-coded entries.

[WindowsCollection] [failure] [DC3.2k8r2-dc1_2k8r2-vcloud_local2] Error subscribing. Transport error code = 6/Connection refused.

The error above indicates that the WinRM service on the target host is not listening on the port that was specified in the Event Source configuration, or that the service is not started.

Transport errors like the one above are typically TCP-level errors or DNS errors. You can use pinging and **nslookup** to resolve the hostname or IP address in the event source, which is typically all that is necessary to determine the root cause of these types of errors.

One of the primary transport errors is the connection to the target traversing a firewall that blocks the connection, or even a local Windows firewall on the system itself. You can be easily determine this by running a **tcdump**:

```
tcpdump host 192.168.12.120 and port 5985
```

This example assumes that the target has a listener on port 5985 (HTTP). If a firewall is blocking access, SYN packets would continually be sent by the Collector to the target without sending any reply back.

The **curl** command is a useful tool for troubleshooting this issue. For example, if clicking the **Test** button on an event source causes the UI to freeze for 30 to 60 seconds, it is probable that:

- Most likely, a firewall is blocking the WinRM port (default ports 5985 for HTTP, 5986 HTTPS).
- Wrong IP address for the target host.
- If the event source address is a hostname, it is resolving to an unknown IP address. Use **nslookup** to verify this on the Collector, and check **/etc/resolve.conf** for the correct DNS server.
- The target system is down. Ping it to be sure.

Winrmconfig script

The winrmconfig script provides the following services:

- A diagnostics tool to discover listener configuration or collection user permission issues.
- A configuration tool to:
  - Initially set up an HTTP or HTTPS listener on the system being collected from (domain controller, domain member system, or a standalone or workgroup system). A listener is required on the system being collected from so that a port can be opened for a system to poll it for records or Security Identifiers (SIDs).
  - Add the necessary local permissions (as shown in the steps below) to a non-administrative Windows account to enable that account to collect events and SIDs from the system the script is run on.
    The permissions to collect events are:
    - Permissions to access the Windows WMI subsystem remotely.
    - Permissions to access the WinRM WMI plugin (which is used to enumerate SIDs via WMI).
    - Membership in the Event Log Readers group
  - Enable the local Windows Network Service account to access the Security Event log. Since Windows Security Events are probably the most critical data that RSA NetWitness collects, this is a critical step. The actual WinRM service in the services panel uses the Network Service account by default. You can check this by right-clicking on the account, selecting **Properties**, and clicking the **Logon** tab.

Run the script in **report** mode to get an indicator of what is failing. You can use the script to diagnose the following commonly seen errors:

401 status code:

If no Kerberos issues were detected in the Collector or Domain Controller issues section, this is typically caused by the collection user not being a member of the Event Log Readers group (the domain-level Event Log Readers group if the target system is a domain controller or the local Event Log Readers group if it's a domain member system or a standalone or workgroup system). The script can fix this no matter which type of system it is. If a hostname is used for the target system instead of an IP address, it MUST be in FQDN format. Similarly, if an IP address is used it MUST resolve to a FQDN.

500 status code:

The typical cause of this error is that an HTTP listener is being used on the target system and the WinRM **AllowUnencrypted** flag was not set. The script can fix this. Another possibility is that there is a corrupted WinRM listener configuration. Look at **/var/log/messages** for a WSMAN fault string that would indicate configuration corruption or clues to the possible source of the issue. If an issue with the **AllowUnencrypted** flag has been ruled out, see the Advanced Errors section.

HTTP Unexpected Error code (0):

This is typically a Windows permissions issue and can be fixed using the **winrmconfig** script.

Test connection failed:Error! Fault Code : s:Receiver Subcode : w:InternalError Reason : The cluster resource is not online.

This occurs when the Security event log ACL has not been modified to allow the Network Service account (which the WinRM Service runs as) read access. See the section on the winconfig Enable mode command to accomplish this.

Error! Could not connect Possible causes: - Windows Collection not running.

This occurs when the Collector connects to the listener initially and finds that it cannot connect either due to a firewall blocking access, a listener is not configured, or the listener is on a different port than the event source is configured for, or maybe a different protocol is configured (HTTP/HTTPS) between the event source and the system's listener. The **winrmconfig** script (described later) can be used to configure a listener.

Other Issues

Another possible problem is that SID enumeration might be failing and not detected, in which case certain fields within certain events would appear in SID format and not user or group names. This can sometimes go unnoticed as events ARE being collected, just not all the detail is conveyed. One way to quickly detect this is to grep the **/var/log/messages** for SID (SID is uppercase). When SID enumeration begins (the default is to run once every 24 hours), it will fail quickly with an access denied message. Again, the **winrmconfig** script can fix this.

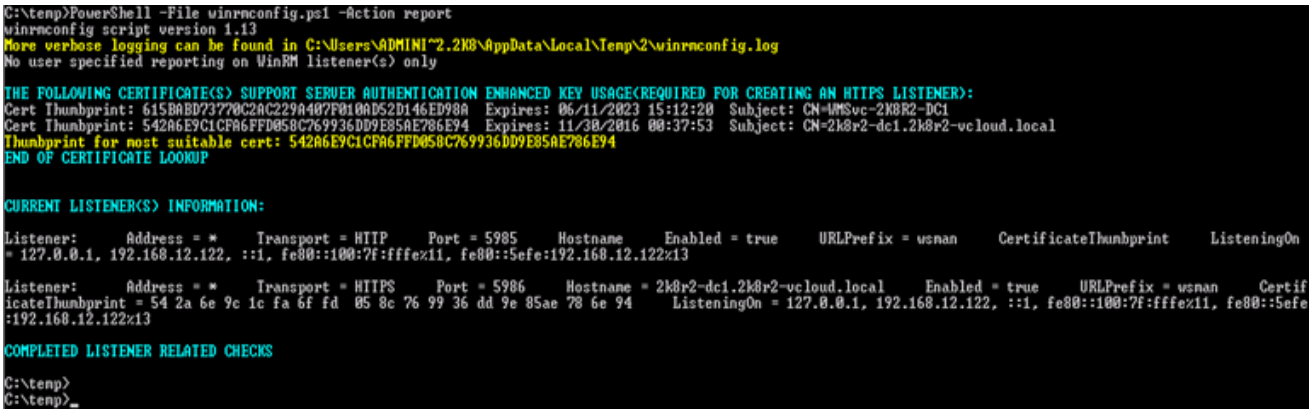
winrmconfig Script Modes

Report mode

This is the troubleshooting mode of the script where the following information is reported back:

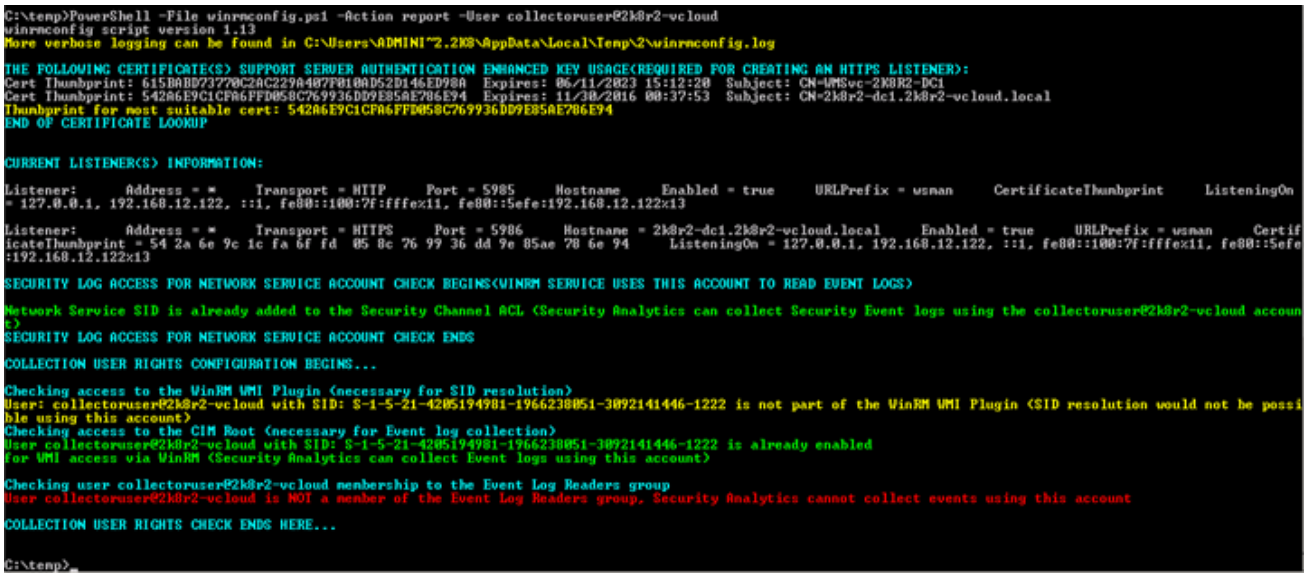
- Details of the available listeners
- All machine certificates that are found that can be used if an HTTPS listener is required.
- The Network Service account permissions to the Security Event Log necessary for collecting
- If a collection user account is passed on the command line:
  - The user's remote access to WMI (necessary to collect events) is reported
  - The user's access to the WinRM WMI plugin is returned (necessary for SID enumeration)
- Event Log Reader group membership is reported.

Note: The user permissions are based on a non-administrative user account that was created (which RSA highly recommends). Not passing the **-user** on the command line will cause the script to return certificate and listener information only.



The Certificate check (shown here) shows that the script has found a viable certificate that could be used if an HTTPS listener is requested. It displays the certificate's thumbprint in yellow, which is a key parameter in creating an HTTPS listener. Also shown in white are the current listeners, the ports the listeners are on, and the thumbprint of the certificate that the HTTPS listener is using. This is shown as an example: it is not typical to see two listeners. In practice, either HTTP or HTTPS is used, so only one listener exists.

Adding **-user** with the collection user's account to the command line adds detail for user permissions while returning the certificate and listener information as well, as shown in the following image.



The Security Event log permissions (access for the Network Service account) are reported on, as well as the collection user's permissions.

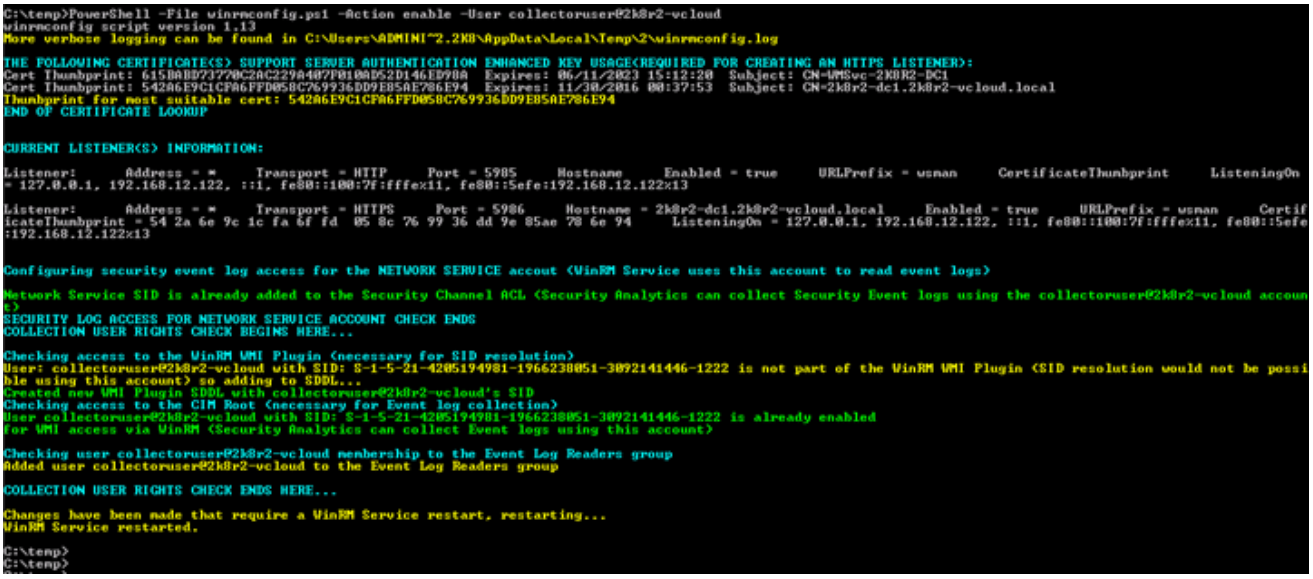
In general, items in yellow are considered warnings, red is an error and green is good. In the report above, the Network Service account DOES have access to the Security Event log (message is in green). However, there is a warning when the check for WinRM WMI plugin access was run. It is a warning because, although this will not stop event collection, SIDs in certain event logs would not be translated to user or group names. Access to the CIM root, which IS necessary for event log collection, was found to be enabled already, so that information displays in green. The final check for Event Log Reader access failed, and that WOULD prevent event logs from being read by the Collector and result in the **HTTP unexpected status code(0)** error.

Enable mode

The **enable** action attempts to fix things. In one pass it can create a listener, enable Security Event Log access and set permissions for a non-administrative account to be able to collect event logs and enumerate SIDs. Doing this is as simple as passing a listener type (HTTP or HTTPS), a customer port number (if required for the listener) and a collection account name for which to enable permissions. In the examples below, the listener type is passed to show listener reporting and enabling. The account name is then passed in to show report and enabling permissions for that account.

Fixing User Permission issues

Running the script in **enable** mode fixes the issues that are discovered, as shown in the following image.



Using **-Action enable** when you run the script, and only specifying the **-user** parameter will cause the tool to show listener and certificate information and will now fix any user permission issues it finds. The output above shows that access to the WinRM WMI plugin is granted and the user is added to the Event Log Readers group. Finally, the WinRM service is restarted to enable the changes.

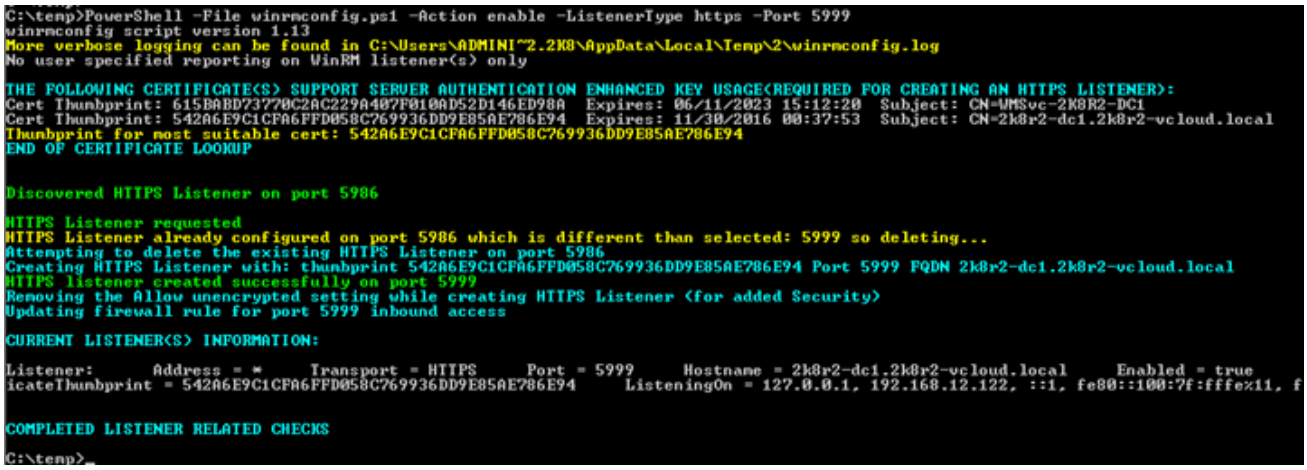
From the output above, collection from this system on any listener shown here, using the collection user specified when running the script, should now work.

Fixing Listener Issues

An HTTP listener is an easy thing to configure in itself. However, the script does more than just creating a listener. For example, an HTTP listener is already on port 5985 but you need to move to HTTPS collection, and IT has opened port 5985 and we don't want to ask them to block that port and now open port 5986. The script can do this by adding **-ListenerType https -Port 5985**. If there is a useable certificate on the system, the script will delete the old HTTP listener and create the HTTPS listener bound to the discovered certificate. Most documentation suggests creating listeners using the **winrm quickconfig** command, but this can fail if the local firewall service is not running. The script bypasses this issue by checking the firewall service state and creating a listener, using manual commands if the firewall service is stopped. In other words, in all cases, the script tries to do the right thing if it is in **enable** mode. It follows what is requested on the command line (listener type, port, suggested certificate thumbprint) to create a listener, and if these conflict with existing listeners, it will delete and re-create the listeners to get to the state that is requested.

> **Note:** While the script creates an HTTP listener, the WinRM **AllowUnencrypted** flag is set to **true**. Without this, a 500 error would be returned to the Collector on connection. (On some versions of Windows a 401 error has also been seen when the flag is not set.)

The following image shows a new HTTPS listener being created on a custom port using the best available certificate it could find. It removes the old HTTPS listener on the default port before creating the new one. Finally, the script detects that the firewall service is running and creates a new firewall rule for the new port.



Note: You cannot create an HTTPS listener on the default HTTP port (5985) and vice versa (an HTTP listener on the default HTTPS port 5986). Windows does not allow that. If you are creating an HTTP listener and planning to migrate to HTTPS later (without any subsequent firewall changes), use a custom port, get the firewall exception for that port, and then use the script to switch between HTTP and HTTPS later.

In the example above, notice that the script can troubleshoot and correct more than 90% of issues with WinRM on the target systems. This is a very simple task, there is no need to understand the underpinnings of the WinRM service and its various layers. Color-coded responses from the script indicate warnings or must-fix situations. Fixing issues can be as simple as switching the action from **report** to **enable**.

The output from the script has been formatted to be easily understood. A more detailed version of the actions that are completed is logged in the path specified when the script has completed:

If you discover any issues while running the script in any mode, submit this log to RSA Support.

Mutual Authentication over HTTPS

Most typical issues with mutual authentication (exporting the CA cert for the certificate bound to the WinRM HTTPS listener and importing to the Collector) are exporting the wrong certificate or picking the wrong format. See the Microsoft WinRM Configuration Guide for a method to simplify exporting the correct CA cert as a PEM format and importing it using the RSA NetWitness UI.

Mass-enabling Systems Using winrmconfig in a GPO

In a domain-based environment during initial WinRM setup, you do not need to manually execute the script on each system (although that is a matter of choice and may depend on the number of systems to be collected from). The script can be pushed as part of a group policy to mass-enable WinRM and configure permissions for a non-administrative user across a number of machines in a domain or a subgroup of a domain (that is, domain controllers only, domain member computers only, computers within certain Organization Units (OUs) in a domain, etc.). A benefit of using GPO is that targeting particular systems can be very flexible using, for example, WMI filtering. See the *Microsoft WinRM Configuration* guide (available for download from SCOL at https://knowledge.rsasecurity.com/) for more information about pushing the script via GPO to many systems.

Advanced Errors

Bookmarks

Sometimes when errors occur, it is not clear if an issue is affecting collection across all channels or just one channel (for example, if the security ACL is not in place and application and system logs can be collected but not security logs). Looking at bookmarks may provide a clue.

The Log Collector tracks where it left off on each channel being collected from using bookmarks. Bookmarks are simply Event Record IDs that correspond to the Record ID's with which Windows tags every event log.

To view bookmarks on a target system:

1. Log onto the system.
2. Start the Event Log Viewer.
3. Right-click on an event log message and select **Properties**.
4. Click **XML view**.

A number of properties, along with record IDs are visible there.

The Log Collector saves the last event log record ID for each channel in a **persistence file**. This method of tracking the last record received is used in a number of collection types.

To find the XML files where bookmarks are stored:

1. From an SSH session logged into the Log Collector or Remote Log Collection service, as **root**, **cd** to:
2. `/var/netwitness/logcollector/runtime/windows/eventsources`
3. An **ls** command will show the XML files where the bookmarks are stored for each system, for example:
   `Domain1.dc1_vcloud_local.xml`

The filename is constructed by concatenating the event category name and the hostname of the target system followed by the domain name (with _ separators instead of .).

Locate the XML filef or the system that the array bounds error is logged against and **watch** it:

```
watch cat Domain1.dc1_vcloud_local.xml
```

The file contains a section called **bookmarks**, one for each of the channels (typically Application, Security, and System). The bookmarks will update when new records are received. If a non-changing event channel number is located by monitoring the error over time, a collection issue with that channel must be occurring.

The **ptime** field in the file can also be monitored. If, for example, the field has not changed in a fairly long period of time, say 5 minutes, it is possible that a collection thread is hung since SID enumeration or collection should not take that long by default.

WinRM-related Connection Errors

WSManFault and general **WS_Management** related errors are protocol errors, which means that they are typically not related to authentication or connection. If they appear in **/var/log/messages**, it is usually safe to assume that authentication has completed (Collection user credentials are valid) but that something else at the WinRM environment or protocol level is wrong, which means looking at the target system itself rather than any configuration on the Collector.

There are no more endpoints available from the endpoint mapper.

Error number: -2147023143 0x800706D9

If this message displays when using **quickconfig** (usually when creating an HTTP listener) or setting **AllowUncrypted** via the WinRM command line, the local firewall is shut down (Windows Firewall Service is stopped) even though it was configured. The local Windows firewall on the target system may be stopped by company policy (for example, if an enterprise firewall is in place). The **winrmconfig** script can bypass these errors. It will detect the firewall state and manually create the listener:

Run the script as shown here:

```
PowerShell -File winrmconfig.ps1 -ListenerType http
```

If a custom port is required for the HTTP listener, run the following:

```
PowerShell -File winrmconfig.ps1 -ListenerType http -Port 5999
```

This can be used where `-Port` allows overriding the default port.

MaxConcurrentOperationsPerUser exceeded

The following error can occur in **/var/log/messages** or as a result of issuing a WinRM command directly on a target system.

```
The WS-Management service cannot process the request. The maximum number of concurrent
operations for this user has been exceeded. Close existing operations for this user, or
raise the quota for this user (0x803381A6)
```

This can be due to:

- Multiple other products (for example, enVision or third party products) are also accessing WinRM on the same system using the same user account as RSA NetWitness.
- The same system is being collected from multiple times by RSA NetWitness (the same event source address is being accessed from different Collectors).

The solution can be to increase the maximum concurrent operations per user via GPO or directly as follows:

```
winrm set winrm/config/Service @{MaxConcurrentOperationsPerUser="400"}
```

and restart the Windows Remote Management service.

WinRM Maximum Sessions Exceeded

The following message has been seen (rarely):

```
The WS-Management service cannot process the request. This user is allowed a maximum number of 5 concurrent shells, which has been exceeded. Close existing shells or raise the quota for this user
```

By default, WinRM allows a maximum of five connections to a remote computer to be active per user. This has been exceeded on sites where other applications are collecting logs via WinRM in parallel with RSA NetWitness (for example, enVision).

To increase the limit, run the following command on the source computer, in which **X** represents the number of connections that you want to allow.

At a command prompt that is opened with elevated user rights, run the following:

```
winrm s winrm/config/winrs @{MaxShellsPerUser="X"}
```

Note: Settings for maximum sessions of up to 100 have been tested successfully in the field.

WSManFault Error number -2144108297 0x803380F7

If **/var/log/messages** contains the following message, OR if issuing a WinRM command on the target system results in this message:

```
WSManFault

Message = The WinRM client cannot process the request. It cannot determine the content type of the HTTP response from the destination computer. The content type is absent or invalid.

Error number: -2144108297 0x803380F7
```

The WinRM client cannot process the request. It cannot determine the content type of the HTTP response from the destination computer. The content type is absent or invalid. See https://support.microsoft.com/en-us/kb/971244 for information about the resolution.

Summary of the issue and resolution:

1. Using Regedit, navigate to:
   `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\HTTP\Parameters`
2. If the hotfix has been applied (running at least Windows 2008 R2), the keys:
   `MaxRequestBytes`
   `MaxFieldLength`
   should already be there.
3. Set both of these keys to **32KB**.
4. Restart the WinRM service.

If the operating system is:

- Windows Server 2008
- Windows Server 2008 Service Pack 2

The hotfix needs to be applied and the values set to 32KB as specified in step 3.

WinRM command returned WSManFault with Error number: -2144108526 0x80338012

```
Message = the client cannot connect to the destination specified in the request.
```

Verify that the service on the destination is running and is accepting requests. Consult the logs and documentation for the WS-Management service running on the destination, most commonly IIS or WinRM. If the destination is the WinRM service, run the following command on the destination to analyze and configure the WinRM service:

```
winrm quickconfig
```

If a GPO was used and it appears to be configured correctly, from an elevated DOS window on the target system, check the listener by running the following command:

```
winrm e winrm/config/listener
```

The following information is displayed:

```
Listener [Source="GPO"]
Address = *
Transport = HTTP
Port = 5985
Hostname
 Enabled = true
 URLPrefix = wsman
CertificateThumbprint
 ListeningOn = null
```

If the Listener is null, it is not listening (a **curltest** from the Collector would have returned `connection refused`).

If `Allow automatic configuration of listeners` is enabled in the policy and an IPv4 filter is set, ensure that the filter includes the target system's IP address, or remove the filter.

grep **/var/log/messages** for the word **Krb5CacheMonitor**.

```
Apr 30 05:44:43 MyLocalLC nw[30413]: [Krb5CacheMonitor] [failure] Failed to fetch Kerberos
TGT for principal : winrmUser@VCLOUD.LOCAL
```

This error could indicate that if collection was working previously, it is possible that the collection user password was changed or the user account was locked out (which can also occur if the password was changed and the cache monitor's attempts to get a token caused the account to be locked out).

```
Apr 30 05:44:43 MyLocalLC nw[30413]: [Krb5CacheMonitor] [info] Fetched Kerberos TGT for
principal : winrmUser@VCLOUD.LOCAL

 Apr 30 05:44:43 MyLocalLC nw[30413]: [Krb5CacheMonitor] [info] Refreshed Kerberos TGT for
principal : winrmUser@VCLOUD.LOCAL
```

The lines above show a successful token update. Both TGT User and Service (HTTP) tickets are placed in the local Kerberos cache. When the Log Collector starts up it exports the cache environment variable.

Collecting from Multiple Domains

The Collector uses Kerberos to retrieve a TGT ticket for the Log Collection user as stated previously. A current limitation is that only one TGT can be stored in the cache at a given time. Although you can create multiple domains with the UI, you should use the same Collection user across those domains. If multiple Log Collection users are used, you will observe odd behavior such as sporadic 401 errors. This is caused by a TGT being retrieved for user A and stored in the local Kerberos cache, then a TGT for user B overwriting the TGT for A. Only the last TGT is used in the connections to all the events sources, and if user B has no access to Site A, these connections will fail. Allowing user A access to the other configured domains, as described in the *Microsoft WinRM Configuration* guide (available from SCOL at https://knowledge.rsasecurity.com/) is the only current workaround.

Truncation of Records

Truncation of records is typically due to either using the wrong locale (see the **Advanced** section of the event source configuration) or lack of permissions to read event logs.

Wrong Locale

The wrong locale means that the locale chosen in the Advanced configuration of an event source is not one that Windows has an installed language pack for. An example of this is configuring RSA NetWitness for the Dutch language solely because the system is configured for Dutch primary language. If events have English language messages, these messages may appear truncated. Leaving the locale to the default allows those messages to pass through to the Collector. The same may also apply if the target system translates logs to local languages. The logs may appear truncated if the locale does not match that language.

Lack of Permission to Read Records

An issue has been reported where RSA NetWitness Windows Security logs were being truncated. Currently, this appears to only occur in Windows 2008 SP2.

One solution is to move the device to Windows 2008 R2. Another solution (described below) allows the issue to be fixed on Windows 2008 SP2. The solution below requires modifying the registry on the device (for example, a domain controller), which requires administrative privileges on the system.

Note: You must take extra care while modifying the Windows registry on any system.

An example of a truncated record is as follows. Note that this issue affects different types of messages not just the Security 5156 message type described here.

```
2012/12/05 15:41:37.999 WIN 192.168.12.122 %NICWIN-4-Security_5156_Microsoft-Windows-
Security-Auditing: Security,rn=3574518 cid=84 eid=4,Wed Dec 05 14:37:09 2012,5156,Microsoft-
Windows-Security-Auditing,,0x8020000000000000,MYSECURITY,12810,,
```

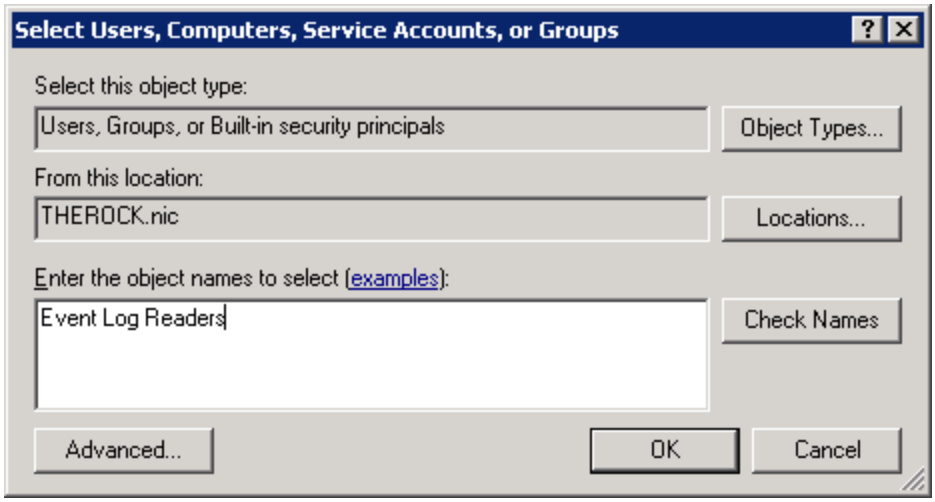
This message typically displays as follows:

```
2012/12/05 15:41:37.999 WIN 192.168.12.122 %NICWIN-4-Security_5156_Microsoft-Windows-
Security-Auditing: Security,rn=3574571 cid=84 eid=4, Wed Dec 05 14:37:09
2012,5156,Microsoft-Windows-Security-Auditing,,Audit Success, MYSECURITY,Filtering Platform
Connection,,The Windows Filtering Platform has allowed a connection. Application
Information: Process ID: 4 Application Name: System Network Information: Direction: Inbound
Source Address: 192.168.12.122 Source Port: 139 Destination Address: 192.168.12.34
Destination Port: 58060 Protocol: 6 Filter Information: Filter Run-Time ID: 0 Layer Name:
Receive/Accept Layer Run-Time ID: 44
```

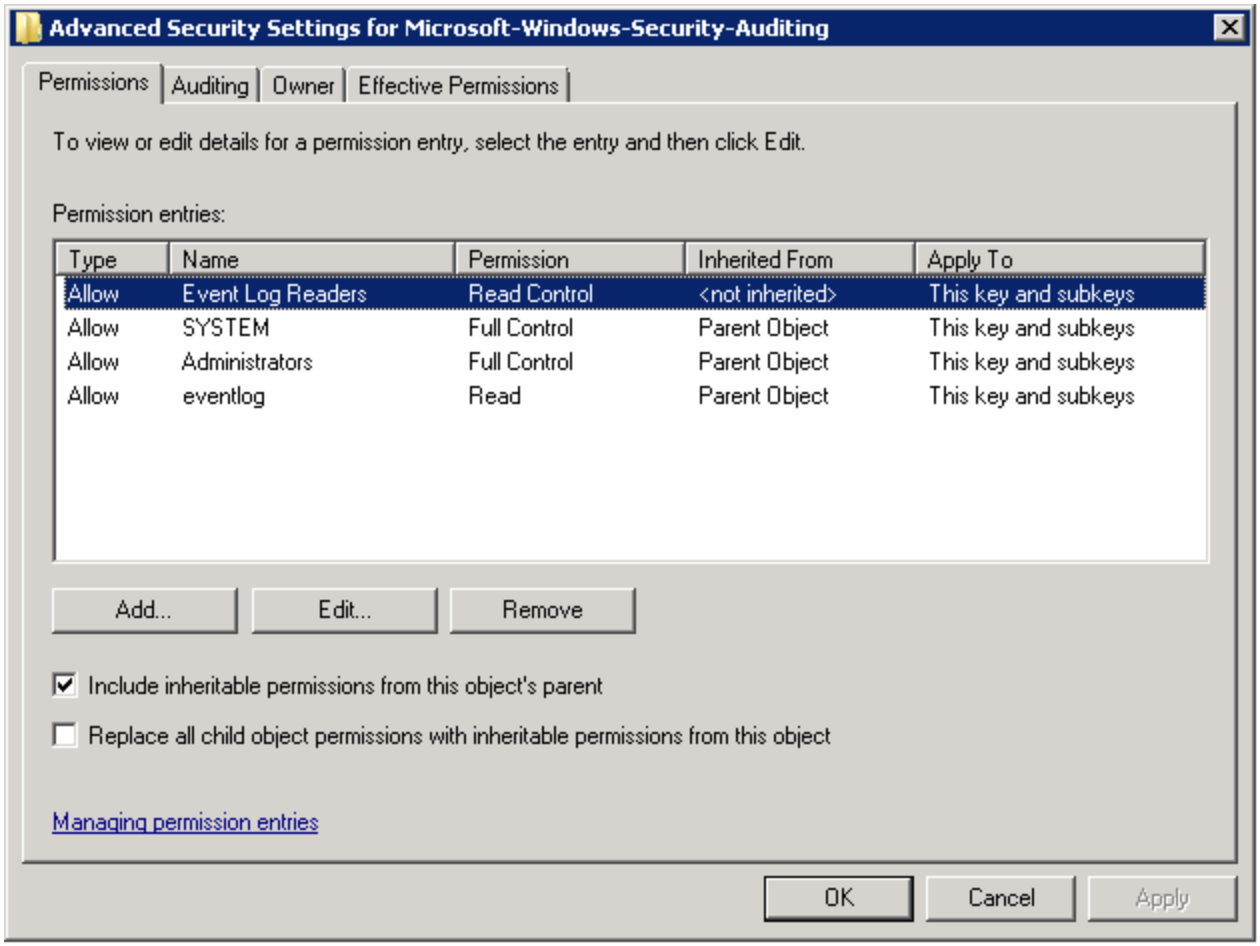
The issue occurs because the Event Log Readers group in the target machine does not have enough permission to allow the event log data to be fully read.

You can change registry permissions on the remote machine to fix permissions issues:

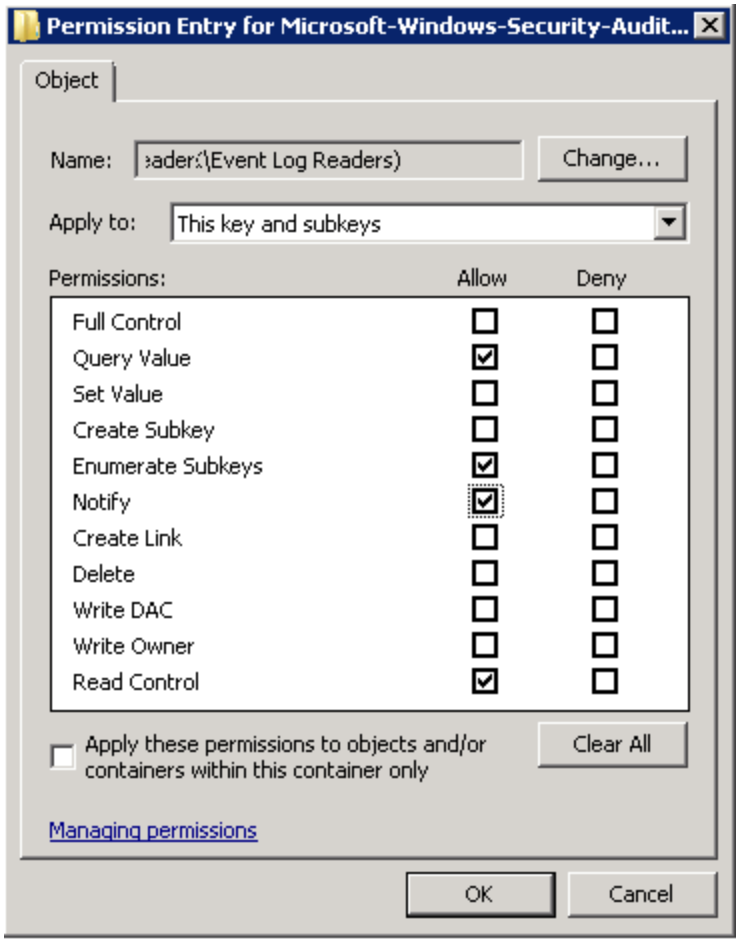
1. Using Regedit, locate the following key:
   `HKEY_LOCAL_MACHINES\System\CurrentControlSet\services\eventlog\Security\Microsoft-Windows-Security-Auditing`
2. Right-click on the key and select **Permissions**.
3. Click the **Add** button in the Permissions for Microsoft-Windows-Security-Auditing panel and enter **Event Log Readers** as shown below.



4. Click **Check Names**. If there are no errors, click **OK**.
5. From the next panel, select **Event Log Readers** and click the **Advanced** button.


Advanced Security Settings for Microsoft-Windows-Security-Auditing
Permissions
Auditing
Owner
Effective Permissions
To view or edit details for a permission entry, select the entry and then click Edit.
Permission entries:
Type
Name
Permission
Inherited From
Apply To
Allow
Event Log Readers
Read Control
<not inherited>
This key and subkeys
Allow
SYSTEM
Full Control
Parent Object
This key and subkeys
Allow
Administrators
Full Control
Parent Object
This key and subkeys
Allow
eventlog
Read
Parent Object
This key and subkeys
Add...
Edit...
Remove
Include inheritable permissions from this object's parent
Replace all child object permissions with inheritable permissions from this object
Managing permission entries
OK
Cancel
Apply

6. Click **Edit**. The Permissions Entry for Microsoft Windows Security audit dialog is displayed.



7. Select the following options:
   Query value
   Enumerate sub keys
   Read control
   Notify

8. Click **OK** and then **Apply**. Click **OK** again to save your updates.

Error retrieving SOAP message due to malformed event XML from the server

An error similar to the following occurs in **/var/log/messages**:

```
Nov 12 22:24:33 SALLDECODER nw[19267]: [WindowsCollection] [failure] [DMZ.192_168_16_11]
Error retrieving SOAP message due to malformed event XML from the server. Current bookmarks
are Application=13230, Security=140906860, System=506518
```

This error is caused by a malformed XML file that was generated when the target system transformed a certain event (a Security Event to be sent to RSA NetWitness).

This can be resolved by installing the following hotfix on all target systems:

http://support.microsoft.com/kb/2956014

You can also resolve this error by excluding that event in the event source or the event category channel string as follows:

```
Security^(4661),System,Application
```

If System or Application logs are not required, omit them from the string .

Caution: Failure to resolve this error will cause data loss!

Collection Occurs with no Events in Investigator

After verifying that Windows collection has been started and checking for events using the **device.ip** or **device.host** meta keys in Investigator, no events were found for a given target host.

In Investigator, when you search for Windows events for a particular system, if the Event Source address is configured as a FQDN, the meta key to look for is **device.host**. If the address is an IP address, look for **device.ip**.

Advanced SOAP Packet Flow

This section provides an overview of the packet flow between Log Collectors and target systems with Negotiate authentication.

The samples that follow have been taken from a **pcap** file created by **tcpdump** on a Log Collector system and loaded into Wireshark using the following command:

```
tcpdump -w output.pcap host 192.168.12.120 and port 5985
```

Sample 1

LC->Target:

```
POST /wsman HTTP/1.1
Host: 192.168.12.120:5985
Accept: */*
Content-Type: application/soap+xml;charset=UTF-8
User-Agent: WS-Management for all
Content-Length: 1580
Expect: 100-continue
```

This sample shows the HTTPS request type (`POST`). It shows a hit on the **/wsman** URL on the host system, with the IP address mentioned in the host header (192.168.12.120 on port 5985). Note that no other information is contained (that is, no credentials). This is expected to fail, which is intentional because the intent is to illicit a response from the target system.

Sample 2

Target->LC

```
HTTP/1.1 401
Server: Microsoft-HTTPAPI/2.0
WWW-Authenticate: Negotiate
WWW-Authenticate: Kerberos
WWW-Authenticate: Basic realm="WSMAN"
Date: Wed, 21 Jan 2015 21:43:54 GMT
Connection: close
Content-Length: 0
```

The target system sends back invalid credentials or an access denied response (an HTTP 401 response).

The target also sent back a list of the authentication types it supports. Note that the default response is typically Negotiate and Kerberos. The response here contains Basic as well, which indicates that someone has probably enabled Basic authentication via a GPO or WinRM set command.

This response enables the Log Collector to analyze if the type of authentication that was selected in the event source configuration (Basic and Kerberos) is supported by the target system.

For example, assume the event source is configured for Basic and the response is as follows :

```
HTTP/1.1 401

Server: Microsoft-HTTPAPI/2.0

WWW-Authenticate: Negotiate

WWW-Authenticate: Kerberos

Date: Wed, 21 Jan 2015 21:43:54 GMT

Connection: close

Content-Length: 0
```

The communications ends here: the POST communication shown in Sample 3 below, with credentials, will not occur.

Sample 3

LC->target

```
POST /wsman HTTP/1.e

Authorization: Negotiate

YIIGMgYJKoZIhvcSAQICAQBuggYhMIIGHaADAgEFoQMCAQ6iBwMFACAAAACjggUvYYIFKzCCBSegAwIBBaEPGw1TUFJBR
1VFLkxPQ0FMoiowKKADAgEDoSEwHxsESFRUUBsXbmhocWFkMDAyLnNwcmFndWUubG9jYWyjggThMIIE3aADAgESoQMCAT
qiggTPBIIEy75mrBz1AQrf8KMrfhe0bsCcXT7DRsZOObb5vE+yi4xTamwFUvZIKkQvEMo2aWdQrLsYzzkxNtnQ0GduIK2
UN8xlP1ufDIX2oZBEbJ85daarUHZF4yOF0odZ+HbFiw1tuhumm0Rd4D6N5TmysCrq+qfb3gXxRP5NYSq4urmZ3S5ruI5b
tMeFvNbELfmQryGU6VHE7svCCEIysL34A37V9GMppMk0gjNBy0YPbAnYwsTvUzsrMui7Pmnl5e4stoYQsQlTaSNpFwoOV
3NTPi5tGrzC7zAgtG5T8bwiW0jP/hOHv/c/GqZ2p6HglGw463pqVn5glqHBtBVN8xvWSLpGupNH5Ll8qkVKbder2j2yBF
aw3rfg5NxkRHWgE/WZuz/Tgy57G7P3S3vfFhJbDsZRC8ncpyGEavVVNbnctuzrd0T9/MmslvoyglETD/sqo6WBd++V+mk
agxiPZAFKFy2gMl8gsA78iy8VbJ+jBBdi40Ez4T67dDwNhRE066msSBNMI0GXYto2V20jV3myTa4sIIfHSN5FHQ+6CdGt
NGjBdz5ZbYkgt7bE2mXiNVk996uI0CN+acI7X6i76yeCdYfeWFph0pDsFPvtoGtyVB/7GTZoVxSS2lV5lAKlQhjKyaboD
9uy9gHkVPDL72z7wVetjGTFBf8PhbIinFlZ3ye+i2+lwkspFTNY8K627Y3KLWmMlmq6Lg9HWD4QyO+H1I6E7PfMo/VOme
JoLztSk9GC90pdHG6yC5B5QPRNL2Y7IUln07vKwveiiuBabBe1HUGj/HXj8huduyAj+pL582g+FtC8unSNbHyGXEh6IoS
+pPp/lkYwdTGnzchTTQ87NSB/DgqchII431JpqGq6DgIvASjqp12eYWDCHajx64PsOeJAtO/qHRCeSAoURByQRcTIF4GB
cA4doTI340572I/t2hNvq/HEJ7xmcuK7wPc/V0AtdqH/CKX3TM+8vWDA9ra6HAe4uVQdvyzKolZ5bCticYWHh4/ofCXgx
J9kQIanEzQSuSDaXdTe8ZNlHOeWpMOkESUyahKhczh4Fi5srRhH8D7j5MVe2sHCDyXJj6vguhoF7vNNz1hr6peNkPKK54
0vEwy1f36MyUPC5KZj+JfRJfGq+u6XoueiP7jQ1LYmpS29tChu6brzkIwg5/Vl3odSi+kAbsY6QevizrYg0D7Np28t8gg
isFqot7U6yf3FXw0LJRz7JOFNJOPoTvCeOZNhUWs92dZUmXDq/GbkjaDErdOOG7MzB0wvcXyTZkdAMy7rnAgG2c+8QLrr
jH/Ko/QllpbkC4vpf00p26Z8PdByGgbA/63PthxHLKr6M8e7xdL6esYTimRsJQQhIDvRNa+vjW/T4gXSneDocuhBfcslf
V7GMpOp3s5Q+bf6gMdcU2b+JYv1iCVK4PkxFaFHdVphEvNXXm1thhgh3yutUsAI5Ym8e8zvB5kLP2iFp7DCm2roe4VLym
UkaUXYNfqR5jGEY9jY/nyuWp9lUAFOmGriGDOB0i55Usu2PibllXjl/poNyeczMgEKhjMAp24mEQ/BMPW2ppcAzGovwLz
c+ip+JST8P+LS+5EPB5VmSDAonZuayCdxzLrXCZSS3PQhLJeOrWUF8rGTac2rxiudUqSB1DCB0aADAgESooHJBIHGIJps
ltNGA9zBeu/oqWAhkQDVik7NgWTMbkdGUeuDX0i32QtVM35uLdoItGfNZ66Fvt3FOj6md7aAvw4F/VL6k+F+rqqpyqCNM
IxsfPZGr1QNUtM5/qrNK0gjrEGncMilH1baRkkc13w9agYQxG7wfkuqATzblKUhy9TBFIP7ms/G8algGZMxogGuxz6hnR
fLZsjH7b+wdaFSyyuZASKdtcauNmQ7iwjg5Nb6UjlJd90px9LLpHg18Yls1jr0+AHbdntxnngd

Host: nhhqad002.sprague.local:5985
```

```
Accept: */*

Content-Type: application/soap+xml;charset=UTF-8

User-Agent: WS-Management for all

Content-Length: 1068

Expect: 100-continue

xml version="1.0" encoding="UTF-8"?>

<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"><s:Header><wsa:Action
s:mustUnderstand="true">http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe

</wsa:Action><wsa:To s:mustUnderstand="true">

http://NHHQDB005.SPRAGUE.LOCAL:5985/wsman</wsa:To><wsman:ResourceURI
s:mustUnderstand="true">http://schemas.microsoft.com/wbem/wsman/1/windows/EventLog

</wsman:ResourceURI><wsa:MessageID s:mustUnderstand="true">uuid:1320ff91-0669-1669-ad46-
1cd55512bcf8</wsa:MessageID><wsa:ReplyTo><wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/
addressing/role/anonymous</wsa:Address></wsa:ReplyTo><wsman:OptionSet>

<wsman:Option Name="ContentFormat">RenderedText</wsman:Option><wsman:Option
Name="IgnoreChannelError">xsi:nil='true'</wsman:Option></wsman:OptionSet></s:Header>

<s:Body><wse:Subscribe>

<wse:Delivery Mode="http://schemas.dmtf.org/wbem/wsman/1/wsman/Pull"/><wsman:Filter
Dialect="http://schemas.microsoft.com/win/2004/08/events/eventquery">&lt;

QueryList&gt;&lt;Query Id='0'&gt;&lt;Select

Path='Application'&gt;*&lt;/Select&gt;&lt;Select

Path='Security'&gt;*&lt;/Select&gt;&lt;Select
Path='System'&gt;*&lt;/Select&gt;&lt;/Query&gt;&lt;/QueryList&gt;</wsman:Filter>

<wsman:SendBookmarks/><wsman:Bookmark><BookmarkList>

<Bookmark Channel="Application" RecordId="37970"/>

<Bookmark Channel="Security" RecordId="330799442"/>

<Bookmark Channel="System" RecordId="302137"/>

</BookmarkList></wsman:Bookmark></wse:Subscribe></s:Body></s:Envelope>
```

This packet contains the HTTP header, stretching from the type (**POST**) down to **Expect: 100**. It is really the same attempt on the **/wsman** URL, but this time with credentials and a WinRM request attached at the end.

The highlighted Authorization header shows that the type is **Negotiate**, which means that you have chosen Kerberos in the event source configuration, rather than **Basic**.

The next section is the large encrypted authorization header that contains primarily the user token and the HTTP token for this host.

Finally, there is the XML-formatted **Subscribe** request. Highlighted within are the important pieces. From top to bottom, these are:

- The **Subscribe** command
- The operation type you are using; `pull` (that is, poll for events) for this packet, as opposed to `push`.
- The **QueryList**, which contains a list of the channels from which you are receiving events. The default list is Security, System, and Application.
- The final section is the list of bookmarks. These are positions in each of the channels in the **QueryList** from which you are receiving information.

Sample 4

LC->target

```
HTTP/1.1 100 Continue
```

If the event source is configured as HTTP (and not HTTPS) and you see an HTTP response of **HTTP/1.1 500** at this point in the trace, it is likely that all credentials are correct, and **AllowUnencrypted** is not set on the host. That is, the following command has not been run:

```
winrm set winrm/config/service @{AllowUnencrypted="true"}
```

If a 401 error is sent back by the target, it is likely that the user is not in the Event Log Readers group, as credential-related 401 errors cause a failure at the very onset of communications.

If collection is working but periodically fails with a 401 error, check the Log Collectors Kerberos ticket refresh thread output. This can be done by grepping **/var/log/messages** for the string **[Krb5CacheMonitor]**. This shows the output from attempts to renew the log collection user tokens. For example:

```
Feb 5 22:05:37 rsalogcolle-0 nw[32364]: [Krb5CacheMonitor] [info] Refreshed Kerberos TGT for
principal : logcollector@2K8R2-VCLOUD.LOCAL
```

A failure message would contain the phrase "Failed to fetch Kerberos TGT for principal," which may mean that you need to check the password in the event category for the user, as that is the number one reason for this scenario. The solution is to verify the password using the **kinit** command, and then re-enter that password into the event category.