

# Windows Event Logs

## **Task 1**      What are event logs?

Per Wikipedia, "**Event logs** record events taking place in the execution of a system to provide an audit trail that can be used to understand the activity of the system and to diagnose problems. They are essential to understand the activities of complex systems, particularly in applications with little user interaction (such as server applications)."

This definition would apply to system administrators, IT technicians, desktop engineers, etc. If the endpoint is experiencing an issue, the event logs can be queried to see clues about what led to the problem. The operating system, by default, writes messages to these logs.

As defenders (blue teamers), there is another use case for event logs. "Combining log file entries from multiple sources can also be useful . *This approach, in combination with statistical analysis, may yield correlations between seemingly unrelated events on different servers.*"

This is where SIEMs (**Security Information and Event Management**) such as Splunk and Elastic come into play.

If you don't know exactly what a SIEM is used for, below is a visual overview of its capabilities.

## SIEM Capabilities

### Main features

- Threat detection
- Investigation
- Time to respond

### Additional features

- Basic security monitoring
- Advanced threat detection
- Forensics & incident response
- Log collection
- Normalisation
- Notifications and alerts
- Security incident detection
- Threat response workflow



Even though accessing a remote machine's event logs is possible, this will not be feasible in a large enterprise environment. Instead, one can view the logs from all the endpoints, appliances, etc., in a SIEM. This will allow you to query the logs from multiple devices instead of manually connecting to a single device to view its logs.

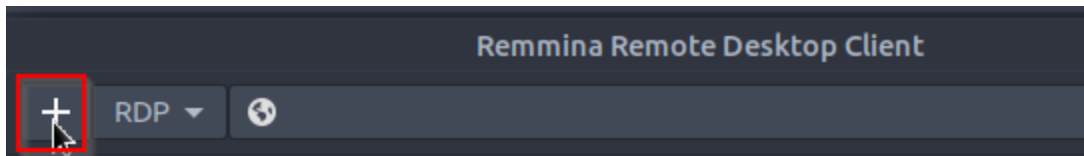
Windows is not the only operating system that uses a logging system. Linux and macOS do as well. For example, the logging system on Linux systems is known as **Syslog**. In this room, though, we're only focusing on the Windows logging system, Windows Event Logs.

## Room Machine

Before moving forward, please deploy the machine.

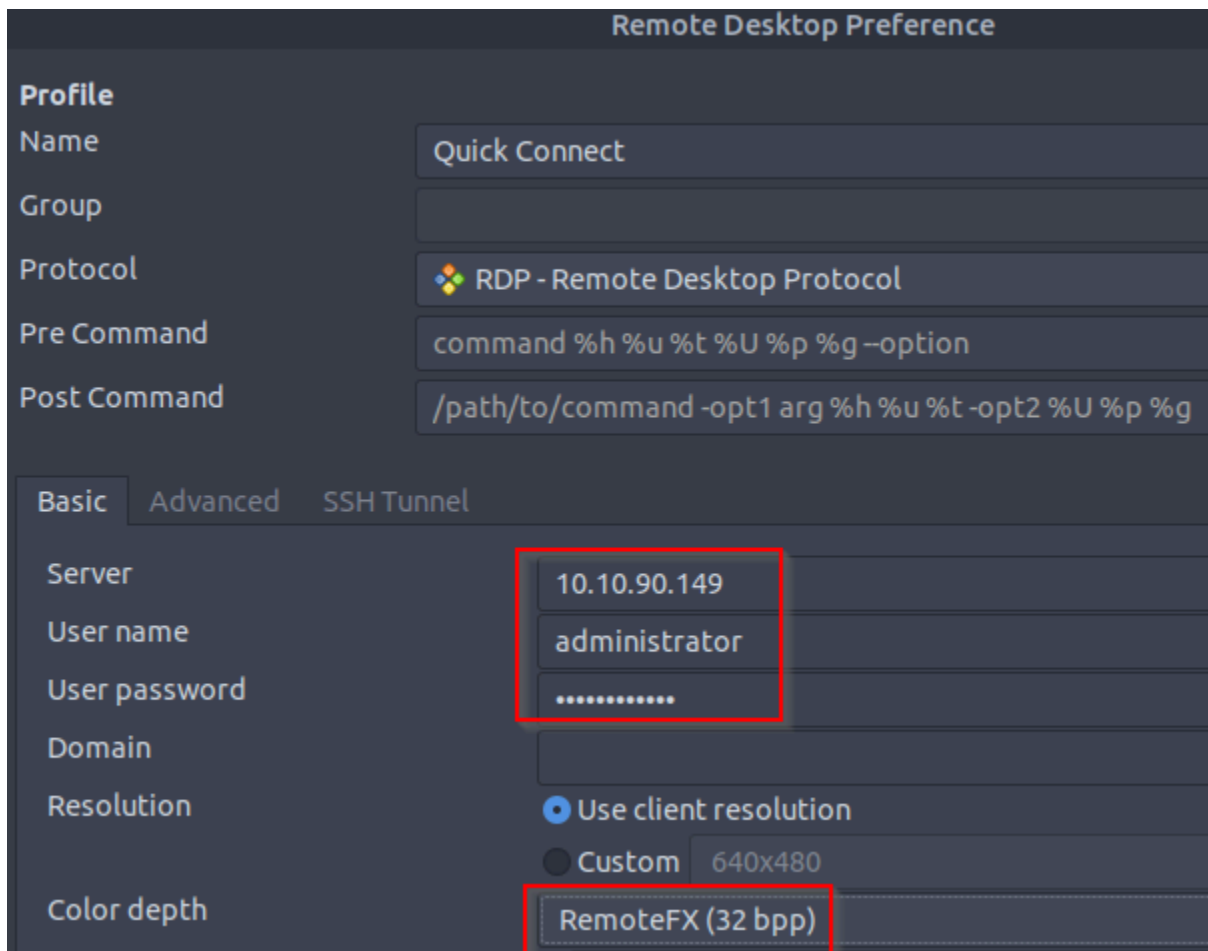
You can use the **AttackBox** and **Remmina** to connect to the remote machine. Make sure the remote machine is deployed before proceeding.

Click on the plus icon, as shown below.



For the **Server**, provide ( ) as the IP address provided to you for the remote machine. The credentials for the user account are:

- User name:
- User password:



Accept the Certificate when prompted, and you should be logged into the remote system now.

**Note :** The virtual machine may take up to 3 minutes to load.

## **Task 2      Event Viewer**

The Windows Event Logs are not text files that can be viewed using a text editor. However, the raw data can be translated into XML using the Windows API. The events in these log files are stored in a proprietary binary format with a .evt or .evtx extension. The log files with the .evtx file extension typically reside in

## Elements of a Windows Event Log

Event logs are crucial for troubleshooting any computer incident and help understand the situation and how to remediate the incident. To get this picture well, you must first understand the format in which the information will be presented. Windows offers a standardized means of relaying this system information.

First, we need to know what elements form event logs in Windows systems. These elements are:

- **System Logs:** Records events associated with the Operating System segments. They may include information about hardware changes, device drivers, system changes, and other activities related to the device.
- **Security Logs:** Records events connected to logon and logoff activities on a device. The system's audit policy specifies the events. The logs are an excellent source for analysts to investigate attempted or successful unauthorized activity.
- **Application Logs :**Records events related to applications installed on a system. The main pieces of information include application errors, events, and warnings.
- **Directory Service Events:** Active Directory changes and activities are recorded in these logs, mainly on domain controllers.
- **File Replication Service Events:** Records events associated with Windows Servers during the sharing of Group Policies and logon scripts to domain controllers, from where they may be accessed by the users through the client servers.
- **DNS Event Logs:** DNS servers use these logs to record domain events and to map out
- **Custom Logs:** Events are logged by applications that require custom data storage. This allows applications to control the log size or attach other parameters, such as ACLs, for security purposes.

Under this categorization, event logs can be further classified into types. Here, types describe the activity that resulted in the event being logged. There are 5 types of events that can be logged, as described in the table below from [docs.microsoft.com](https://docs.microsoft.com).

The following table describes the five event types used in event logging.

Event type	Description
Error	An event that indicates a significant problem such as loss of data or loss of functionality. For example, if a service fails to load during startup, an Error event is logged.
Warning	An event that is not necessarily significant, but may indicate a possible future problem. For example, when disk space is low, a Warning event is logged. If an application can recover from an event without loss of functionality or data, it can generally classify the event as a Warning event.
Information	An event that describes the successful operation of an application, driver, or service. For example, when a network driver loads successfully, it may be appropriate to log an Information event. Note that it is generally inappropriate for a desktop application to log an event each time it starts.
Success Audit	An event that records an audited security access attempt that is successful. For example, a user's successful attempt to log on to the system is logged as a Success Audit event.
Failure Audit	An event that records an audited security access attempt that fails. For example, if a user tries to access a network drive and fails, the attempt is logged as a Failure Audit event.

There are three main ways of accessing these event logs within a Windows system:

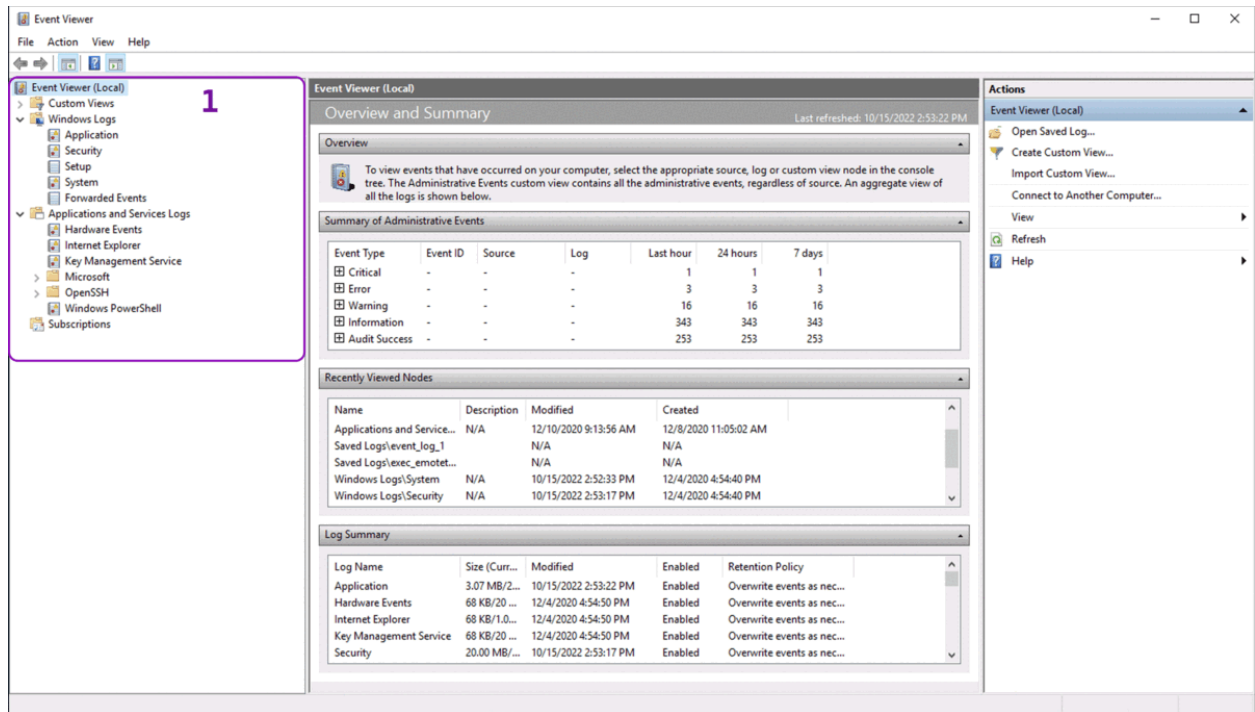
1. **Event Viewer** (GUI-based application)
2. **Weventutil.exe** (command-line tool)
3. **Get-WinEvent** (PowerShell cmdlet)

## Event Viewer

In any Windows system, the Event Viewer, a **Microsoft Management Console (MMC)** snap-in, can be launched by simply right-clicking the Windows icon in the taskbar and selecting **Event Viewer**. For the savvy sysadmins that use the CLI much of their day, Event Viewer can be launched by typing `eventvwr`. It is a GUI-based application that allows you to interact quickly with and analyze logs.

Event Viewer has three panes.

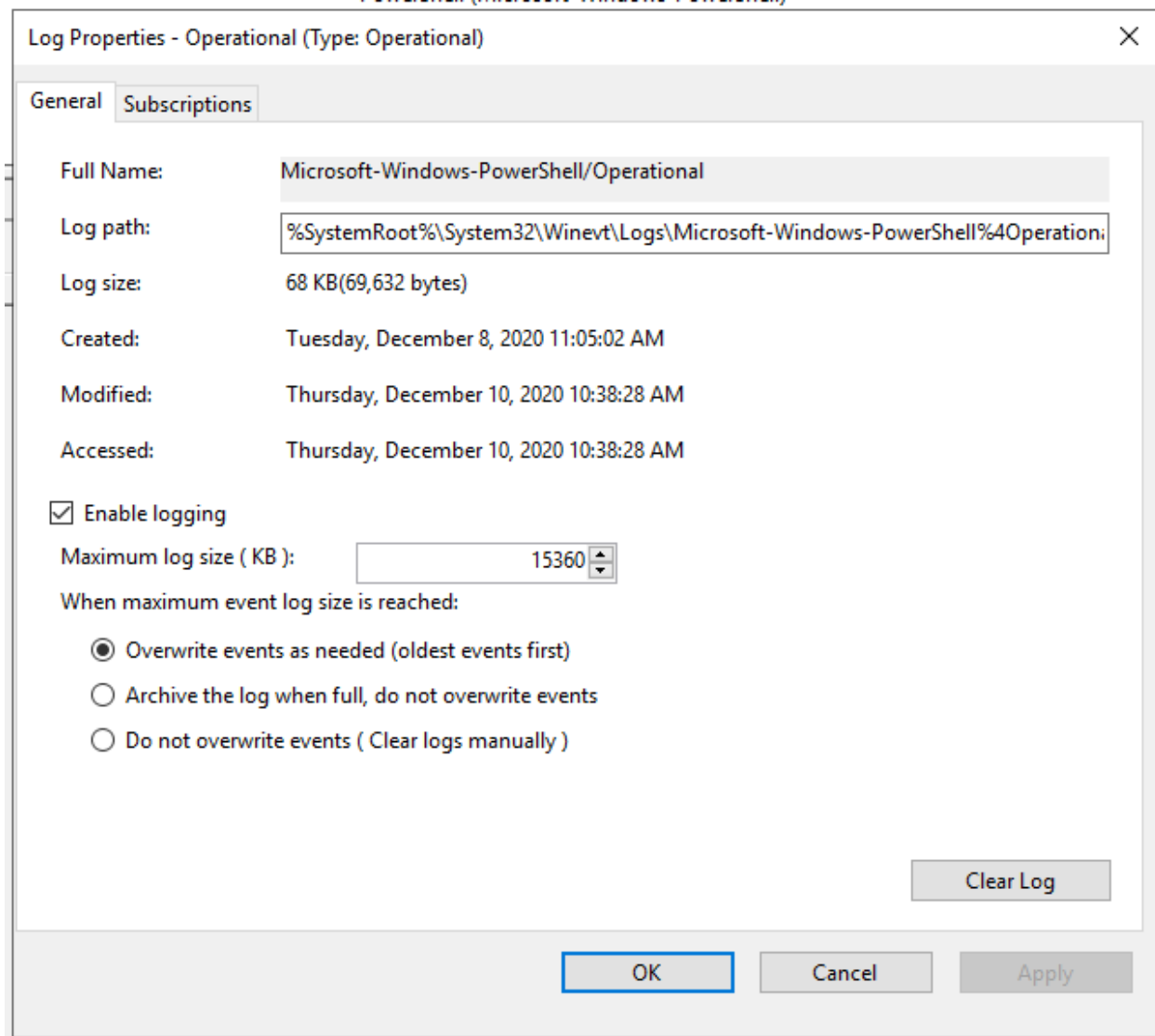
1. The pane on the left provides a hierarchical tree listing of the event log providers.
2. The pane in the middle will display a general overview and summary of the events specific to a selected provider.
3. The pane on the right is the actions pane.



The standard logs we had earlier defined on the left pane are visible under **Windows Logs**.

The following section is the **Applications and Services Logs**. Expand this section and drill down on **Operational**. PowerShell will log operations from the engine, providers, and cmdlets to the Windows event log.

Right-click on **Operational** then **Properties**.



Within **Properties**, you see the log location, log size, and when it was created, modified, and last accessed. Within the Properties window, you can also see the maximum set log size and what action to take once the criteria are met. This concept is known as log rotation. These are discussions held with corporations of various sizes. How long does it take to keep logs, and when it's permissible to overwrite them with new data.

Lastly, notice the **Clear Log** button at the bottom right. There are legitimate reasons to use this button, such as during security maintenance, but adversaries will likely attempt to clear the logs to go undetected. **Note:** This is not the only method to remove the event logs for any given event provider.

Focus your attention on the middle pane. Remember from previous descriptions that this pane will display the events specific to a selected provider. In this case, **PowerShell/Operational**.

Operational Number of events: 44				
Level	Date and Time	Source	Event ID	Task Category
Information	12/10/2020 10:38:01 AM	PowerShell (Microsoft-Windows-PowerShell)	4103	Executing Pipeline

From the above image, notice the event provider's name and the number of events logged. In this case, there are 44 events logged. You might see a different number. No worries, though. Each column of the pane presents a particular type of information as described below:

- **Level:** Highlights the log recorded type based on the identified event types specified earlier. In this case, the log is labeled as **Information**.
- **Date and Time:** Highlights the time at which the event was logged.
- **Source:** The name of the software that logs the event is identified. From the above image, the source is PowerShell.
- **Event ID:** This is a predefined numerical value that maps to a specific operation or event based on the log source. This makes Event IDs not unique, so in the above image is related to Executing Pipeline but will have an entirely different meaning in another event log.
- **Task Category:** Highlights the Event Category. This entry will help you organize events so the Event Viewer can filter them. The event source defines this column.

The middle pane has a split view. More information is displayed in the bottom half of the middle pane for any event you click on.

This section has two tabs: **General** and **Details**.

- General is the default view, and the rendered data is displayed.
- The Details view has two options: Friendly view and XML view.

Below is a snippet of the General view.



Event 4103, PowerShell (Microsoft-Windows-PowerShell)

General Details

CommandInvocation(PSConsoleHostReadLine): "PSConsoleHostReadLine"

Context:

- Severity = Informational
- Host Name = ConsoleHost
- Host Version = 5.1.17763.592
- Host ID = 3154c106-cb32-4b2f-8aee-ba629a2e1ce8
- Host Application = C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

Log Name: Microsoft-Windows-PowerShell/Operational

Source: PowerShell (Microsoft-Wind

Logged: 12/10/2020 10:38:01 AM

Event ID: 4103

Task Category: Executing Pipeline

Level: Information

Keywords: None

User: WIN-100UJBNP9G7\Admini

Computer: WIN-100UJBNP9G7

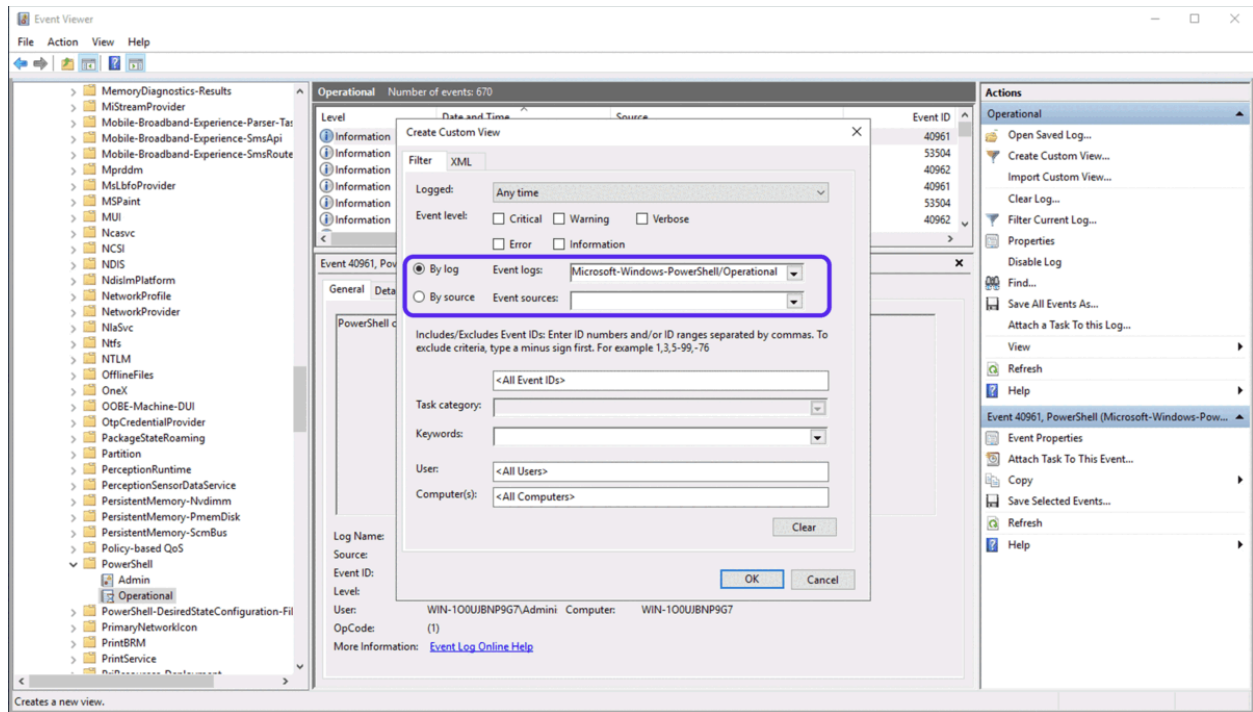
OpCode: To be used when operation i

More Information: [Event Log Online Help](#)

Lastly, take a look at the **Actions** pane. Several options are available, but we'll only focus on a few. Please examine all the actions that can be performed at your leisure if you're unfamiliar with MMC snap-ins.

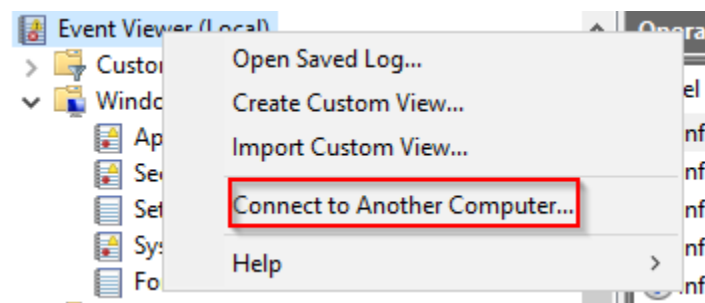
As you should have noticed, you can open a saved log within the Actions pane. This is useful if the remote machine can't be accessed. The logs can be provided to the analyst. You will perform this action a little later.

The **Create Custom View** and **Filter Current Log** are nearly identical. The only difference between the 2 is that the ☐ and ☐ radio buttons are greyed out in **Filter Current Log**. What is the reason for that? The filter you can make with this specific action only relates to the current log. Hence no reason for 'by log' or 'by source' to be enabled.



Why are these actions beneficial? Say, for instance, you don't want all the events associated with PowerShell/Operational cluttering all the real estate in the pane. Maybe you're only interested in 4104 events. That is possible with these two actions.

To view event logs from another computer, right-click



That will conclude the general overview of the Event Viewer—time to become familiar with the tool.

**Note:** Don't forget to deploy the machine for this room before proceeding. Give the room about 3 minutes to load fully.

**Answer the questions below**

**What is the Event ID for the earliest recorded event?**

40961

**Filter on Event ID 4104. What was the 2nd command executed in the PowerShell session?**

*whoami*

**What is the Task Category for Event ID 4104?**

*Execute a Remote Command*

**Analyze the Windows PowerShell log. What is the Task Category for Event ID 800?**

*Pipeline Execution Details*

### **Task 3      wevtutil.exe**

Ok, you played around with Event Viewer. Imagine you have to sit there and manually sift through hundreds or even thousands of events (even after filtering the log). Not fun. It would be nice if you could write scripts to do this work for you. We will explore some tools that will allow you to query event logs via the command line and/or PowerShell.

Let's look at **wevtutil.exe** first. Per Microsoft, the wevtutil.exe tool "enables you to retrieve information about event logs and publishers. You can also use this command to install and uninstall event manifests, to run queries, and to export, archive, and clear logs."

As with any tool, access its help files to find out how to run the tool. An example of a command to do this is

Powershell - wevtutil.exe

PS> C:\Users\Administrator> wevtutil.exe /?

Windows Events Commandline Utility.

Enables you to retrieve information about event logs and publishers, install and uninstall event manifests, run queries, and export, archive and clear logs.

Usage:

You can use either the short (for example, ep /uni) or long (for example, enum-publishers /unicode) version of the command and option names. Commands, options and option values are not case-sensitive.

Variables are noted in all upper-case.

wevtutil COMMAND [ARGUMENT [ARGUMENT] ...] [/OPTION:VALUE [/OPTION:VALUE] ...]

## Commands:

el   enum-logs	List log names.
gl   get-log	Get log configuration information.
sl   set-log	Modify configuration of a log.
ep   enum-publishers	List event publishers.
gp   get-publisher	Get publisher configuration information.
im   install-manifest	Install event publishers and logs from manifest.
um   uninstall-manifest	Uninstall event publishers and logs from manifest.
qe   query-events	Query events from a log or log file.
gli   get-log-info	Get log status information.
epl   export-log	Export a log.
al   archive-log	Archive an exported log.
cl   clear-log	Clear a log.

From the above snippet, under **Usage**, you are provided a brief example of how to use the tool. In this example, **(enum-publishers)** is used. This is a **command** for wevtutil.exe.

Below, we can find the **Common options** that can be used with Windows Events Utility.

Powershell - wevtutil.exe

Common Options:

**/[r | remote]:VALUE**

If specified, run the command on a remote computer. VALUE is the remote computer name.

Options /im and /um do not support remote operations.

**/[u | username]:VALUE**

Specify a different user to log on to the remote computer. VALUE is a user name in the form of domain\user or user. Only applicable when option /r is specified.

**/[p | password]:VALUE**

Password for the specified user. If not specified, or if VALUE is "", the user will be prompted to enter a password. Only applicable when the /u option is specified.

**/[a | authentication]:[Default|Negotiate|Kerberos|NTLM]**

Authentication type for connecting to remote computer. The default is Negotiate.

**/[uni | unicode]:[true|false]**

Display output in Unicode. If true, then output is in Unicode.

To learn more about a specific command, type the following:

wevtutil COMMAND /?

Notice at the bottom of the above snapshot, . This will provide additional information specific to a command. We can use it to get more information on the command **(query-events)**.

Powershell - wevtutil.exe

PS> C:\Users\Administrator> wevtutil qe /?

Read events from an event log, log file or using a structured query.

Usage:

wevtutil {qe | query-events} [/OPTION:VALUE [/OPTION:VALUE]...]

Look over the information within the help menu to fully understand how to use this command.

Ok, great! You have enough information to use this tool—time to answer some questions. It is always recommended to look into the tool and its related information at your own leisure.

**Note** : You can get more information about using this tool further but visiting the online help documentation [docs.microsoft.com](https://docs.microsoft.com).

**Answer the questions below**

**How many log names are in the machine?**

Open powershell and use the command `wevtutil.exe el | Measure-Object`

*1071*

**What event files would be read when using the query-events command?**

*event log, log file, structured query*

**What option would you use to provide a path to a log file?**

*/f:true*

**What is the VALUE for /q?**

*Xpath query*

**What is the log name?**

*Application*

**What is the /rd option for?**

*Event read direction*

**What is the /c option for?**

*Maximum number of events to read*

#### **Task 4      Get-WinEvent**

## **Get-WinEvent**

On to the next tool. This is a PowerShell cmdlet called **Get-WinEvent**. Per Microsoft, the Get-WinEvent cmdlet "gets events from event logs and event tracing log files on local and remote computers." It provides information on event logs and event log providers. Additionally, you can combine numerous events from multiple sources into a single command and filter using XPath queries, structured XML queries, and hash table queries.

**Note:** The **Get-WinEvent** cmdlet replaces the **Get-EventLog** cmdlet.

As with any new tool, it's good practice to read the Get-Help documentation to become acquainted with its capabilities. Please refer to the Get-Help information online at [docs.microsoft.com](https://docs.microsoft.com).

Let us look at a couple of examples of how to use Get-WinEvent, as supported by the documentation. Some tasks might require some PowerShell-fu, while others don't. Even if your PowerShell-fu is not up to par, fret not; each example has a detailed explanation of the commands/cmdlets used.

### **Example 1: Get all logs from a computer**

Here, we are obtaining all event logs locally, and the list starts with classic logs first, followed by new Windows Event logs. It is possible to have a log's **RecordCount** be zero or null.

Powershell - Get-WinEvent Logs

```
Get-WinEvent -ListLog *
```

```
LogMode    MaximumSizeInBytes RecordCount LogName
```

```

-----
Circular      15532032    14500 Application
Circular      1052672     117 Azure Information Protection
Circular      1052672     3015 CxAudioSvcLog
Circular      20971520      ForwardedEvents
Circular      20971520      0 HardwareEvents

```

## Example 2: Get event log providers and log names

The command here will result in the event log providers and their associated logs. The **Name** is the provider, and **LogLinks** is the log that is written to.

Powershell - Get-WinEvent Providers

```
Get-WinEvent -ListProvider *
```

```
Name      : .NET Runtime
```

```
LogLinks  : {Application}
```

```
Opcodes   : {}
```

```
Tasks     : {}
```

```
Name      : .NET Runtime Optimization Service
```

```
LogLinks  : {Application}
```

```
Opcodes   : {}
```

```
Tasks     : {}
```

### Example 3: Log filtering

Log filtering allows you to select events from an event log. We can filter event logs using the **Where-Object** cmdlet as follows:

Powershell - Get-WinEvent Filters

```
PS C:\Users\Administrator> Get-WinEvent -LogName Application | Where-Object {
$_.ProviderName -Match 'WLMS' }
```

ProviderName: WLMS

TimeCreated	Id	Level	DisplayName	Message
12/21/2020 4:23:47 AM	100	Information		
12/18/2020 3:18:57 PM	100	Information		
12/15/2020 8:50:22 AM	100	Information		
12/15/2020 8:18:34 AM	100	Information		
12/15/2020 7:48:34 AM	100	Information		
12/14/2020 6:42:18 PM	100	Information		
12/14/2020 6:12:18 PM	100	Information		
12/14/2020 5:39:08 PM	100	Information		
12/14/2020 5:09:08 PM	100	Information		

**Tip:** If you are ever working on a Windows evaluation virtual machine that is cut off from the Internet eventually, it will shut down every hour. ;^)

When working with large event logs, per Microsoft, it's inefficient to send objects down the pipeline to a `ForEach-Object` command. The use of the Get-WinEvent cmdlet's **FilterHashtable**



parameter is recommended to filter event logs. We can achieve the same results as above by running the following command:

Powershell - Get-WinEvent Filters

```
Get-WinEvent -FilterHashtable @{  
  
    LogName='Application'  
  
    ProviderName='WLMS'  
  
}
```

The syntax of a hash table is as follows:

Hash Table Syntax

```
@{ <name> = <value>; [<name> = <value> ] ...}
```

Guidelines for defining a hash table are:

- Begin the hash table with an @ sign.
- Enclose the hash table in braces {}
- Enter one or more key-value pairs for the content of the hash table.
- Use an equal sign (=) to separate each key from its value.

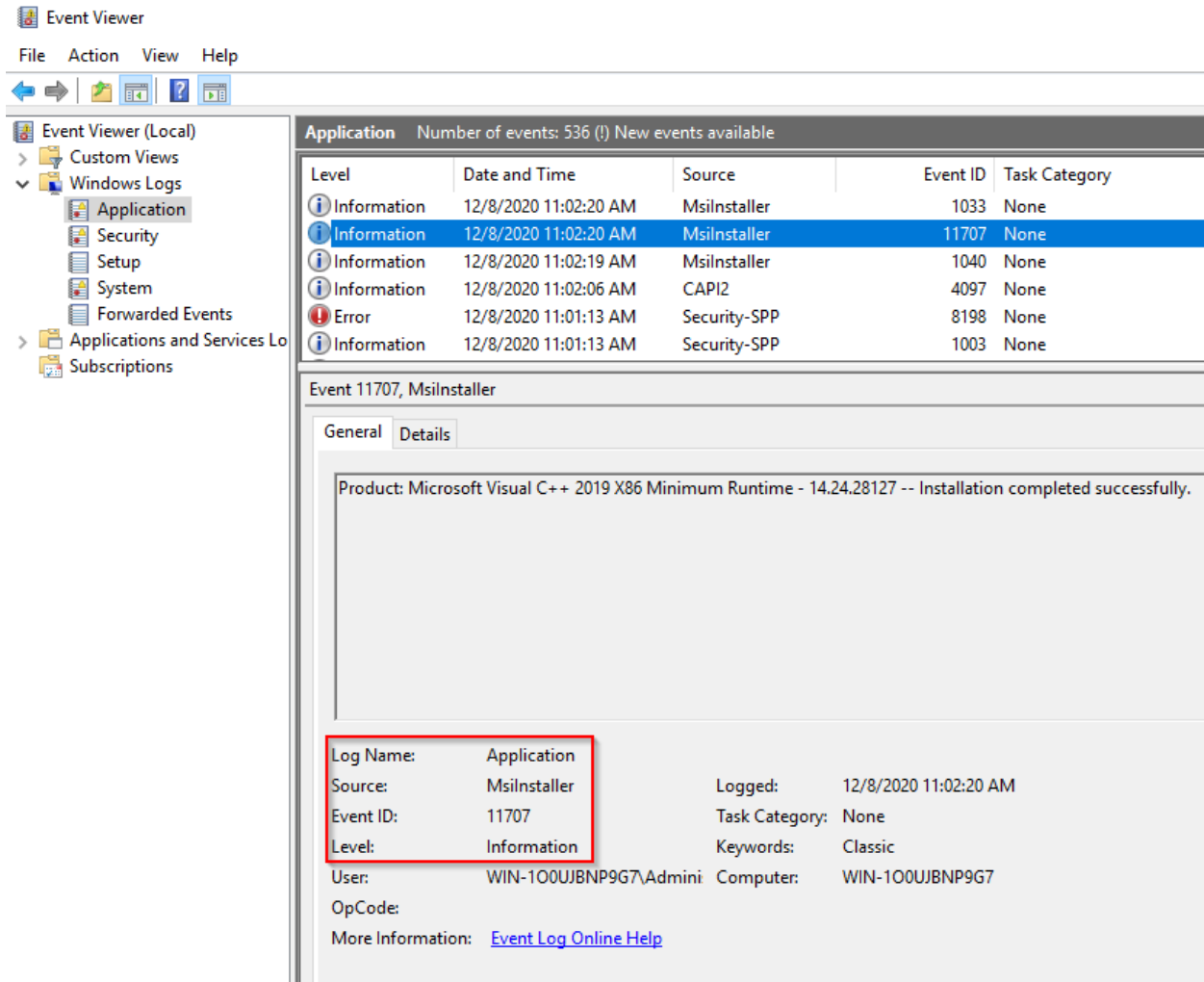
**Note:** You don't need to use a semicolon if you separate each key/value with a new line, as in the screenshot above for the -FilterHashtable for .

Below is a table that displays the accepted key/value pairs for the Get-WinEvent FilterHashtable parameter.

The following table displays the key names, data types, and whether wildcard characters are accepted for a data value.

Key name	Value data type	Accepts wildcard characters?
LogName	<String[]>	Yes
ProviderName	<String[]>	Yes
Path	<String[]>	No
Keywords	<Long[]>	No
ID	<Int32[]>	No
Level	<Int32[]>	No
StartTime	<DateTime>	No
EndTime	<DateTime>	No
UserID	<SID>	No
Data	<String[]>	No
<named-data>	<String[]>	No

When building a query with a hash table, Microsoft recommends making the hash table one key-value pair at a time. Event Viewer can provide quick information on what you need to build your hash table.



Based on this information, the hash table will look as follows:

```
1 Get-WinEvent -FilterHashtable @{
2     LogName='Application'
3     ProviderName='MsiInstaller'
4     ID=11707
5 }
```

For more information on creating Get-WinEvent queries with FilterHashtable, check the official Microsoft documentation [docs.microsoft.com](https://docs.microsoft.com).

Since we're on the topic of Get-WinEvent and FilterHashtable, here is a command that you might find helpful (shared by [@mubix](#)):

Powershell - Get-WinEvent Filters

```
Get-WinEvent -FilterHashtable @{LogName='Microsoft-Windows-PowerShell/Operational';
ID=4104} | Select-Object -Property Message | Select-String -Pattern 'SecureString'
```

You can read more about creating hash tables in general [docs.microsoft.com](https://docs.microsoft.com).

### **Answer the questions below**

**Execute the command from Example 1 (as is). What are the names of the logs related to OpenSSH?**

```
Get-WinEvent -ListLog * | Out-String -Stream | Select-String -Pattern "OpenSSH"
```

*OpenSSH/Admin,OpenSSH/Operational*

**Execute the command from Example 8. Instead of the string \*Policy\* search for \*PowerShell\*. What is the name of the 3rd log provider?**

```
Get-WinEvent -ListProvider *Powershell* | Format-Table -AutoSize -Wrap
```

*Microsoft-Windows-PowerShell-DesiredStateConfiguration-FileDownloadManager*

**Execute the command from Example 9. Use Microsoft-Windows-PowerShell as the log provider. How many event ids are displayed for this event provider?**

```
(Get-WinEvent -ListProvider Microsoft-Windows-Powershell).Events |  
Format-Table Id, Description | Measure-Object
```

*192*

**How do you specify the number of events to display?**

*-MaxEvents*

**When using the FilterHashtable parameter and filtering by level, what is the value for Informational?**

*4*

### **Task 5      XPath Queries**

Now we will examine filtering events with **XPath**. The W3C created XPath, or **XML Path Language** in full, to provide a standard syntax and semantics for addressing parts of an XML document and manipulating strings, numbers, and booleans. The Windows Event Log supports a subset of [XPath 1.0](https://www.w3.org/TR/xpath-1.0/).

Below is an example XPath query along with its explanation:






### XPath Query

// The following query selects all events from the channel or log file where the severity level is less than or equal to 3 and the event occurred in the last 24 hour period.

XPath Query: `*[System[(Level <= 3) and TimeCreated[timediff(@SystemTime) <= 86400000]]]`

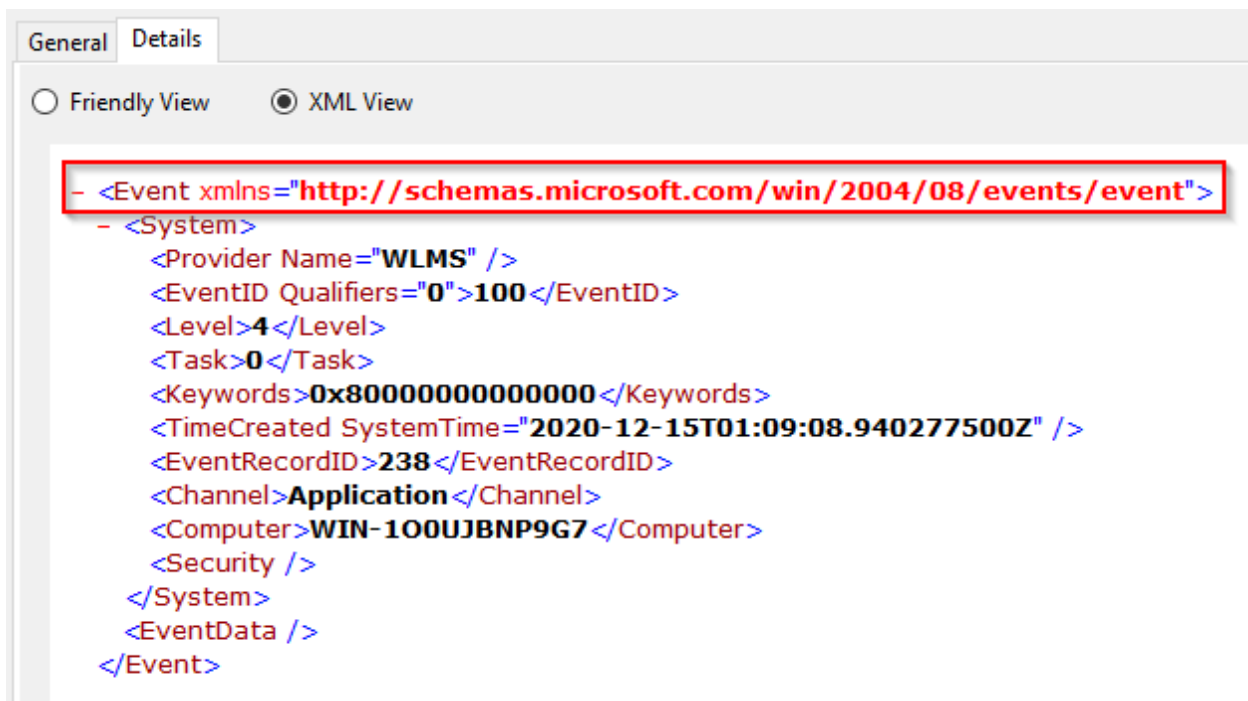
Based on [docs.microsoft.com](https://docs.microsoft.com/en-us/windows/win32/eventlog), an XPath event query starts with '\*' or 'Event'. The above code block confirms this. But how do we construct the rest of the query? Luckily the Event Viewer can help us with that.

Let's create an XPath query for the same event from the previous section. Note that both wevtutil and Get-WinEvent support XPath queries as event filters.

Filtered: Log: Application; Source: ; Event ID: 100. Number of events: 7				
Level	Date and Time	Source	Event ID	Task Category
 Information	12/14/2020 5:09:08 PM	WLMS	100	None
 Information	12/14/2020 5:39:08 PM	WLMS	100	None
 Information	12/14/2020 6:12:18 PM	WLMS	100	None
 Information	12/14/2020 6:42:18 PM	WLMS	100	None
 Information	12/15/2020 7:48:34 AM	WLMS	100	None

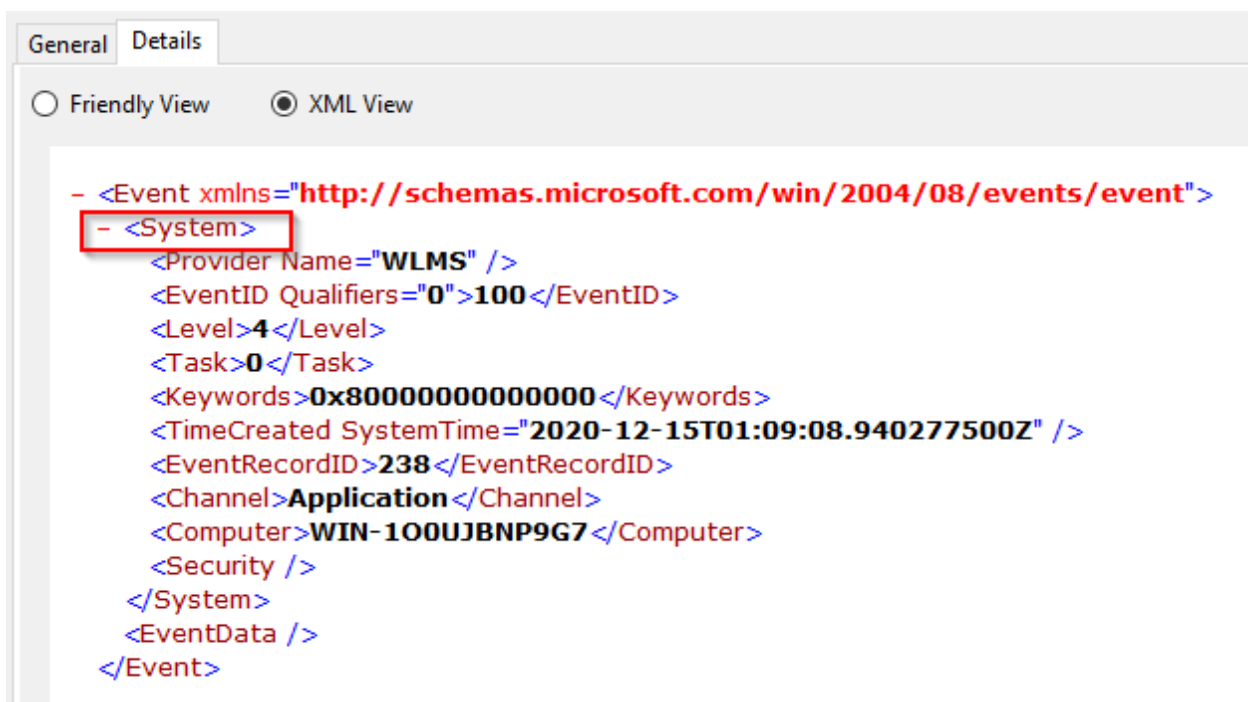
Draw your attention to the bottom half of the middle pane. In the Event Viewer section, the Details tab was briefly touched on. Now you'll see how the information in this section can be useful.

Click on the ☐ tab and select the ☐ radio button. Don't worry if the log details you are viewing are slightly different. The point is understanding how to use the XML View to construct a valid XPath query.



The first tag is the starting point. This can either be an `<Event>` or the word `<System>`.

The command so far looks like this:

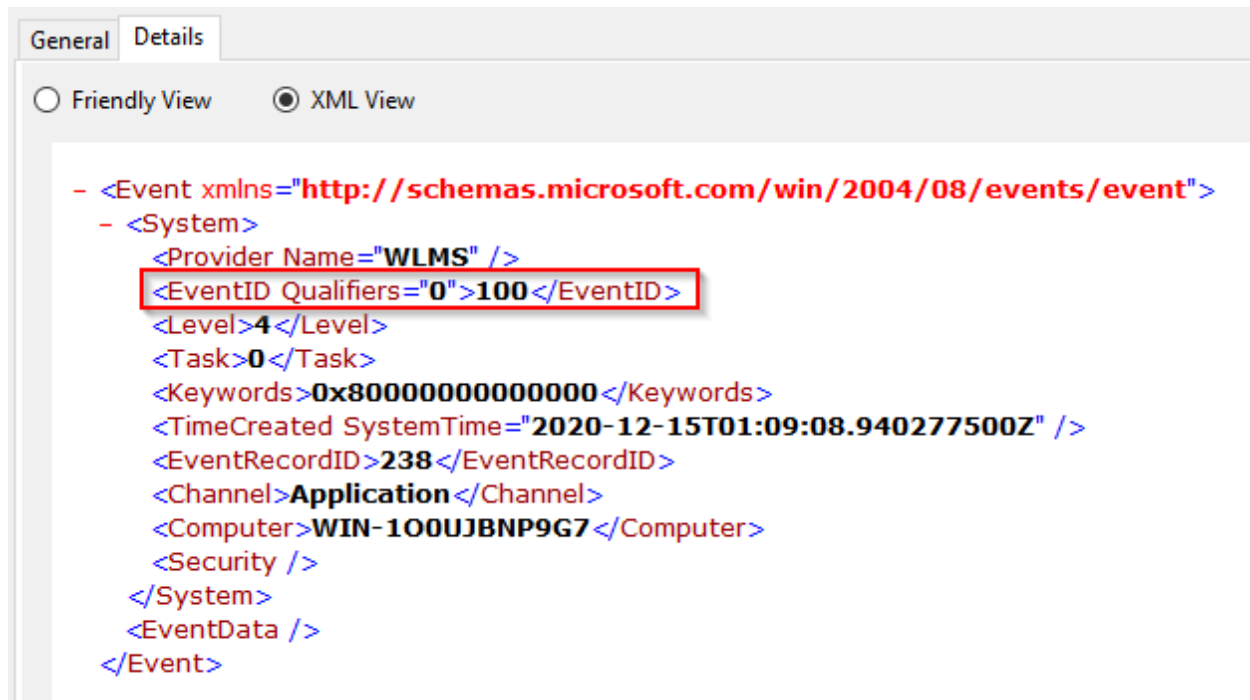


Now we work our way down the XML tree. The next tag is .

Let's add that. Now our command is:

**Note:** Its best practice to explicitly use the keyword `<EventID Qualifiers="0">` but you can use an `<EventID>` instead as with the `<EventID>` keyword. The query is still valid.

The **Event ID** is **100**. Let's plug that into the command.



Our command now is:

XPath Query Powershell

```
PS C:\Users\Administrator> Get-WinEvent -LogName Application -FilterXPath
'*/System/EventID=100'
```

ProviderName: WLMS

TimeCreated                      Id LevelDisplayName Message

12/21/2020 4:23:47 AM	100 Information
12/18/2020 3:18:57 PM	100 Information
12/15/2020 8:50:22 AM	100 Information
12/15/2020 8:18:34 AM	100 Information
12/15/2020 7:48:34 AM	100 Information
12/14/2020 6:42:18 PM	100 Information
12/14/2020 6:12:18 PM	100 Information
12/14/2020 5:39:08 PM	100 Information
12/14/2020 5:09:08 PM	100 Information

When using wevtutil.exe and XPath to query for the same event log and ID, this is our result:

XPath Query using Wevtutil.exe

```
C:\Users\Administrator>wevtutil.exe qe Application /q:*/System[EventID=100] /f:text /c:1
```

Event[0]:

Log Name: Application

Source: WLMS

Date: 2020-12-14T17:09:08.940

Event ID: 100

Task: None

Level: Information

Opcode: Info

Keyword: Classic



User: N/A

User Name: N/A

Computer: WIN-100UJBNP9G7

Description:

N/A

**Note:** 2 additional parameters were used in the above command. This was done to retrieve just 1 event and for it not to contain any XML tags.

If you want to query a different element, such as `System/Provider`, the syntax will be different. To filter on the provider, we need to use the `@Name` attribute of `Provider`.

The XPath query is:

XPath Query for Provider

```
PS C:\Users\Administrator> Get-WinEvent -LogName Application -FilterXPath
'*/System/Provider[@Name="WLMS"]'
```

ProviderName: WLMS

TimeCreated	Id	Level	DisplayName	Message
-------------	----	-------	-------------	---------

12/21/2020 4:23:47 AM	100	Information		
12/18/2020 3:18:57 PM	100	Information		
12/15/2020 8:50:22 AM	100	Information		
12/15/2020 8:48:34 AM	101	Information		
12/15/2020 8:18:34 AM	100	Information		
12/15/2020 7:48:34 AM	100	Information		

12/14/2020 7:12:18 PM	101 Information
12/14/2020 6:42:18 PM	100 Information
12/14/2020 6:12:18 PM	100 Information
12/14/2020 6:09:09 PM	101 Information
12/14/2020 5:39:08 PM	100 Information
12/14/2020 5:09:08 PM	100 Information

What if you want to combine 2 queries? Is this possible? The answer is yes.

Let's build this query based on the screenshot above. The Provider Name is **WLMS**, and based on the output, there are **2 Event IDs**.

This time we only want to query for events with **Event ID 101**.

The XPath query would be

XPath Two Queries

```
PS C:\Users\Administrator> Get-WinEvent -LogName Application -FilterXPath
'*/System/Provider[@Name="WLMS"]'
```

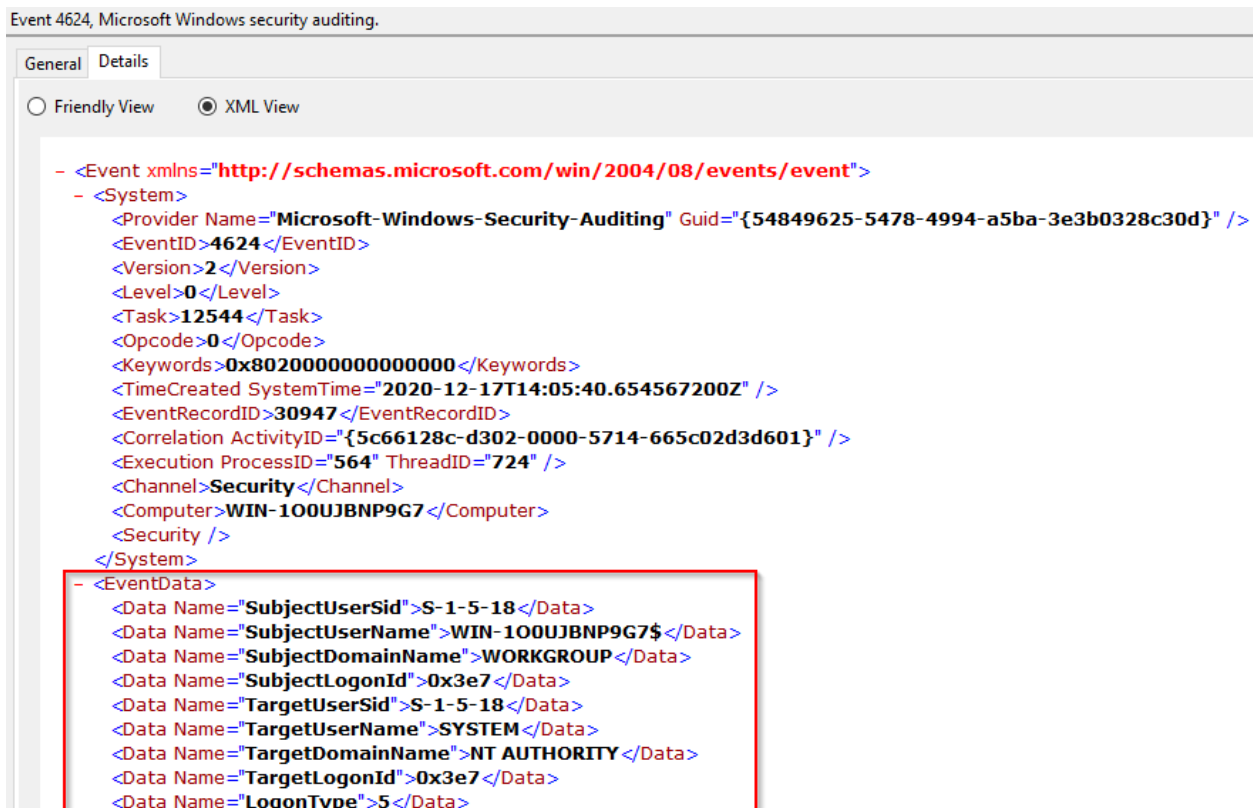
ProviderName: WLMS

TimeCreated	Id LevelDisplayName Message
-----	-- -----
12/15/2020 8:48:34 AM	101 Information
12/14/2020 7:12:18 PM	101 Information
12/14/2020 6:09:09 PM	101 Information

Lastly, let's discuss how to create XPath queries for elements within . The query will be slightly different.

**Note:** The EventData element doesn't always contain information.

Below is the XML View of the event for which we will build our XPath query.



We will build the query for . In this case, that will be System. The XPath query would be

XPath Query for TargetUserName

```
PS C:\Users\Administrator> Get-WinEvent -LogName Security -FilterXPath
'*/EventData/Data[@Name="TargetUserName"]="System"' -MaxEvents 1
```

ProviderName: Microsoft-Windows-Security-Auditing

TimeCreated	Id	Level	DisplayName	Message
-------------	----	-------	-------------	---------

-----	-----	-----	-----	-----
-------	-------	-------	-------	-------

12/21/2020 10:50:26 AM	4624	Information		An account was successfully logged on...
------------------------	------	-------------	--	--

**Note:** The `MaxEvents` parameter was used, and it was set to 1. This will return just 1 event.

At this point, you have enough knowledge to create XPath queries for **wevtutil.exe** or **Get-WinEvent**. To further this knowledge, I suggest reading the official Microsoft XPath Reference [docs.microsoft.com](https://docs.microsoft.com/en-us/windows/win32/weset/xpath).

**Answer the questions below**

**Using the knowledge gained on Get-WinEvent and XPath, what is the query to find WLMS events with a System Time of 2020-12-15T01:09:08.940277500Z?**

*Get-WinEvent -LogName Application -FilterXPath '\*/System/Provider[@Name="WLMS"] and \*/System/TimeCreated[@SystemTime="2020-12-15T01:09:08.940277500Z"]'*

**Using Get-WinEvent and XPath, what is the query to find a user named Sam with an Logon Event ID of 4720?**

*Get-WinEvent -LogName Security -FilterXPath '\*/EventData/Data[@Name="TargetUserName"]="Sam" and \*/System/EventID=4720'*

**Based on the previous query, how many results are returned?**

2

**Based on the output from the question #2, what is Message?**

*A user account was created*

**Still working with Sam as the user, what time was Event ID 4724 recorded? (MM/DD/YYYY H:MM:SS [AM/PM])**

Change the EventId

*12/17/2020 1:57:14 PM*

**What is the Provider Name?**

### **Task 6**      **Event IDs**

When it comes to monitoring and hunting, you need to know what you are looking for. There are a large number of event IDs in use. This section is aimed at assisting you with this task. There are plenty of blogs, writeups, etc., on this topic. A few resources will be shared in this section. Please note this is not an exhaustive list.

First on the list is [The Windows Logging Cheat Sheet \(Windows 7 - Windows 2012\)](#). The last version update is October 2016, but it's still a good resource. The document covers a few things that need to be enabled and configured and what event IDs to look for based on different categories, such as Accounts, Processes, Log Clear, etc.

**WEvtUtil:** Use this utility to query your logs

- a. WevtUtil qe Security – query the Security Log for events
  - i. Lots of flags here so read help “WevtUtil -?”
  - ii. /c:5 = Read 5 events
  - iii. /rd:true = newest events first
  - iv. /f:text = format text, also can do XML
- b. **Success & Failed Logons** - WevtUtil qe Security /q:"\*[System[(EventID=4624 or EventID=4625)]]" /c:5 /rd:true /f:text >Parsed\%computername%\_Logon\_Events\_Win7.log
- c. **User Account Change** - WevtUtil qe Security /q:"\*[System[(EventID=4738)]]" /c:5 /rd:true /f:text >Parsed\R\_%computername%\_User\_Account\_Change\_Win7.log
- d. **New Service Installed** - WevtUtil qe Security /q:"\*[System[(EventID=7045)]]" /c:5 /rd:true /f:text >Parsed\R\_%computername%\_New\_Service\_Installed\_Win7.log
- e. **User Account Changes** - wevtutil qe Security /q:"\*[System[(EventID=4725 or EventID=4722 or EventID=4723 or EventID=4724 or EventID=4726 or EventID=4767)]]" /c:10 /f:text

Above is a snippet from the cheatsheet. Want to detect if a new service was installed? Look for **Event ID 7045** within the **System Log**.

Next is [Spotting the Adversary with Windows Event Log Monitoring](#). This NSA resource is also a bit outdated but good enough to build upon your foundation. The document covers some concepts touched on in this room and beyond. You must click on [here](#) to download the resource.

## 4.5 Windows Firewall

If client workstations are taking advantage of the built-in host-based Windows Firewall, then there is value in collecting events to track the firewall status. For example, if the firewall state changes from on to off, then that log should be collected. Normal users should not be modifying the firewall rules of their local machine.

	ID	Level	Event Log	Event Source
Firewall Rule Add	2004	Informational	Microsoft-Windows-Windows Firewall With Advanced Security/Firewall	Microsoft-Windows-Windows Firewall With Advanced Security
Firewall Rule Change	2005	Informational	Microsoft-Windows-Windows Firewall With Advanced Security/Firewall	Microsoft-Windows-Windows Firewall With Advanced Security
Firewall Rules Deleted	2006, 2033	Informational	Microsoft-Windows-Windows Firewall With Advanced Security/Firewall	Microsoft-Windows-Windows Firewall With Advanced Security
Firewall Failed to load Group Policy	2009	Error	Microsoft-Windows-Windows Firewall With Advanced Security/Firewall	Microsoft-Windows-Windows Firewall With Advanced Security

Table 6: Firewall Events

The above events for the listed versions of the Windows operating system are only applicable to modifications of the local firewall settings.

Above is a snippet from the document. Maybe you want to monitor if a firewall rule was deleted from the host. That is **Event ID 2006/2033**.

Where else can we get a list of event IDs to monitor/hunt for?[MITRE ATT&CK!](#)

If you are unfamiliar with **MITRE** or **MITRE ATT&CK**, I suggest you check out the [MITRE Room](#).

Let's look at ATT&CK ID [T1098](#) (Account Manipulation). Each ATT&CK ID will contain a section sharing tips to mitigate the technique and detection tips.

### Detection

Collect events that correlate with changes to account objects and/or permissions on systems and the domain, such as event IDs 4738, 4728 and 4670 unusual systems. Especially flag events where the subject and target accounts differ<sup>(11)</sup> or that include additional flags such as changing a password w

Monitor for use of credentials at unusual times or to unusual systems or services. This may also correlate with other suspicious activity.

Monitor for unusual permissions changes that may indicate excessively broad permissions being granted to compromised accounts.

The last two resources are from **Microsoft**:

- [Events to Monitor](#) (Best Practices for Securing Active Directory)
- [The Windows 10 and Windows Server 2016 Security Auditing and Monitoring Reference](#) (a comprehensive list [over 700 pages])

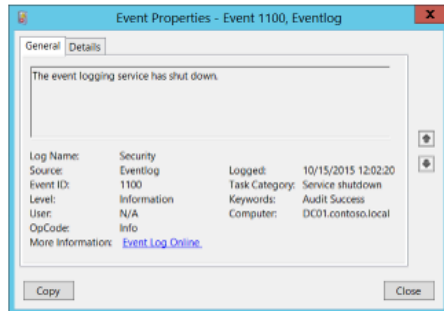
## Other Events

Events in this section generate automatically and are enabled by default.

### Events List:

- [1100\(S\)](#): The event logging service has shut down.
- [1102\(S\)](#): The audit log was cleared.
- [1104\(S\)](#): The security log is now full.
- [1105\(S\)](#): Event log automatic backup.
- [1108\(S\)](#): The event logging service encountered an error while processing an incoming event published from %1

[1100\(S\)](#): The event logging service has shut down.



### Event Description:

This event generates every time Windows Event Log service has shut down. It also generates during normal system shutdown. This event doesn't generate during emergency system reset.

**Note** For recommendations, see [Security Monitoring Recommendations](#) for this event.

### Event XML:

```
- <Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
- <System>
  <Provider Name="Microsoft-Windows-Eventlog" Guid="{fc65ddd8-d6ef-4962-83d5-6e5cfe9ce148}" />
  <EventID>1100</EventID>
  <Version>0</Version>
  <Level>4</Level>
  <Task>103</Task>
  <Opcode>0</Opcode>
  <Keywords>0x4020000000000000</Keywords>
  <TimeCreated SystemTime="2015-10-15T07:02:20.010585400Z" />
  <EventRecordID>1048124</EventRecordID>
  <Correlation />
  <Execution ProcessID="820" ThreadID="964" />
  <Channel>Security</Channel>
  <Computer>DC01.contoso.local</Computer>
  <Security />
</System>
- <UserData>
  <ServiceShutdown xmlns="http://manifests.microsoft.com/win/2004/08/windows/eventlog" />
```

**Note:** Some events will not be generated by default, and certain features will need to be enabled/configured on the endpoint, such as PowerShell logging. This feature can be enabled via **Group Policy** or the **Registry**.

Windows PowerShell			
Select an item to view its description.	Setting	State	Comment
	Turn on Module Logging	Enabled	No
	Turn on PowerShell Script Block Logging	Enabled	No
	Turn on Script Execution	Enabled	No
	Turn on PowerShell Transcription	Enabled	No
	Set the default source path for Update-Help	Not configured	No

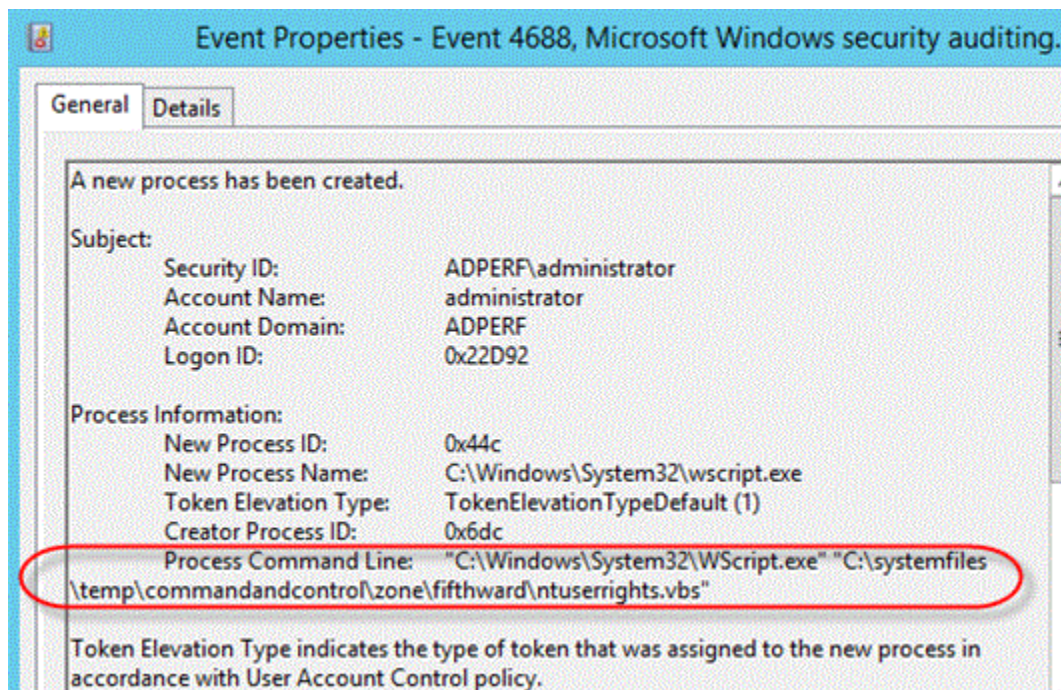
Some resources to provide more information about enabling this feature, along with its associated event IDs:

- [About Logging Windows](#)
- [Greater Visibility Through PowerShell Logging](#)
- [Configure PowerShell logging to see PowerShell anomalies in Splunk UBA](#)

- EventCode = 4103
- EventCode = 4104
- EventCode = 4688 and Process\_Name contains PowerShell
- EventCode = 7045 and Process\_Name contains PowerShell

Another feature to enable/configure is **Audit Process Creation**, which will generate **event ID 4688**. This will allow **command-line process auditing**. This setting is NOT enabled in the virtual machine but feel free to enable it and observe the events generated after executing some commands.

Audit Process Creation			
Select an item to view its description.	Setting	State	Comment
	Include command line in process creation events	Not configured	No



To read more about this feature, refer to [docs.microsoft.com](https://docs.microsoft.com). The steps to test the configuration are at the bottom of the document.



## Try This: Explore command line process auditing

1. Enable **Audit Process Creation** events and ensure the Advance Audit Policy configuration is not overwritten
2. Create a script that will generate some events of interest and execute the script. Observe the events. The script used to generate the event in the lesson looked like this:

```
mkdir c:\systemfiles\temp\commandandcontrol\zone\fifthward
copy \\192.168.1.254\c$\hidden c:\systemfiles\temp\hidden\commandandcontrol\zone\fifthward
start C:\systemfiles\temp\hidden\commandandcontrol\zone\fifthward\ntuserrights.vbs
del c:\systemfiles\temp\*. * /Q
```

Copy

3. Enable the command line process auditing
4. Execute the same script as before and observe the events

To conclude this section, it will be reiterated that this is not an exhaustive list. There are countless blogs, writeups, threat intel reports, etc., on this topic.

To effectively monitor and detect, you need to know what to look for (as mentioned earlier).

### **Task 7**      **Putting theory into practice**

**Note** : To successfully answer the questions below, you may need to search online for more information.

The next scenarios/questions are based on the external event log file titled \_\_\_\_\_ found on the Desktop. You can use any of the aforementioned tools to answer the questions below.

Scenario 1 (Questions 1 & 2): The server admins have made numerous complaints to Management regarding PowerShell being blocked in the environment. Management finally approved the usage of PowerShell within the environment. Visibility is now needed to ensure there are no gaps in coverage. You researched this topic: what logs to look at, what event IDs to monitor, etc. You enabled PowerShell logging on a test machine and had a colleague execute various commands.

Scenario 2 (Questions 3 & 4): The Security Team is using Event Logs more. They want to ensure they can monitor if event logs are cleared. You assigned a colleague to execute this action.

Scenario 3 (Questions 5, 6 & 7): The threat intel team shared its research on **Emotet** . They advised searching for event ID 4104 and the text "ScriptBlockText" within the EventData element. Find the encoded PowerShell payload.

Scenario 4 (Questions 8 & 9) : A report came in that an intern was suspected of running unusual commands on her machine, such as enumerating members of the Administrators group. A senior analyst suggested searching for ". Confirm the suspicion.

**Answer the questions below**

**What event ID is to detect a PowerShell downgrade attack?**

400

**What is the Date and Time this attack took place? (MM/DD/YYYY H:MM:SS [AM/PM])**

Filtered: Log: file://C:\Users\Administrator\Desktop\merged.evbx Source: ; Event ID: 400. Number of events: 113				
Level	Date and Time	Source	Event ID	Task Category
Information	12/18/2020 7:50:33 AM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/18/2020 7:48:45 AM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/18/2020 7:48:45 AM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/18/2020 7:48:43 AM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/18/2020 7:48:42 AM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/18/2020 7:43:49 AM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/18/2020 7:43:28 AM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/18/2020 7:43:28 AM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/18/2020 7:42:45 AM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/18/2020 7:42:45 AM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/18/2020 7:42:29 AM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/18/2020 7:42:29 AM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/17/2020 2:02:34 PM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/17/2020 12:21:17 PM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/17/2020 12:20:57 PM	PowerShell (PowerShell)	400	Engine Lifecycle
Information	12/16/2020 4:46:39 PM	PowerShell (PowerShell)	400	Engine Lifecycle

12/18/2020 7:50:33 AM

**A Log clear event was recorded. What is the 'Event Record ID'?**

27736

**What is the name of the computer?**

PC01.example.corp

**What is the name of the first variable within the PowerShell command?**

\$Va5w3n8

**What is the Date and Time this attack took place? (MM/DD/YYYY H:MM:SS [AM/PM])**

8/25/2020 10:09:28 PM

**What is the Execution Process ID?**

6620

**What is the Group Security ID of the group she enumerated?**

*S-1-5-32-544*

**What is the event ID?**

*4799*

## **Task 8      Conclusion**

In this room, we covered Windows Event Logs, what they are, and how to query them using various tools and techniques.

We also briefly discussed various features within Windows that you need to enable/configure to log additional events to gain visibility into those processes/features that are turned off by default.

The information covered in this room will serve as a primer for other rooms covering [Windows Internals](#), [Sysmon](#), and various [SIEM](#) tools.

I'll end this room by providing additional reading material:

- [EVTX Attack Samples](#) (a few were used in this room)
- [PowerShell <3 the Blue Team](#)
- [Tampering with Windows Event Tracing: Background, Offense, and Defense](#)