

Querying Exported Data for Process or Service Anomalies



Liam Cleary

Microsoft MVP and Microsoft Certified Trainer at SharePlicity

@helloitsliam | www.helloitsliam.com



Overview

Goal: Analyze Collected Data for Anomalies

- Analyzing and converting event logs
- Writing queries for event logs
- Importing event log entries into a database
- Query event log data within the database



Analyzing Event Logs for Anomalies



Preparing Event Logs for Analyzing



Launch Event Viewer

- Click "Save all events as"
- Set the name and file extension as *.evtx
- Right click "Event viewer" root and choose "Open saved log"
- Browse to the saved log and open



Preparing Event Logs for Analyzing

```
# Set the Log to Export and Output Location and Filename
$log = "Security"
$logDate = Get-Date -format yyyyMMddHHmm
$path = "C:\PSFolder\"
$file = ".evtx"
$output = $path + $log + "-" + $logDate + $file

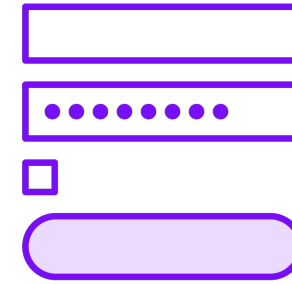
# Export the Log
wevtutil epl $log $output
```



Common Anomaly Categories



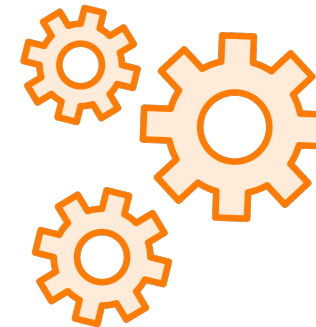
Suspicious account activity and behavior



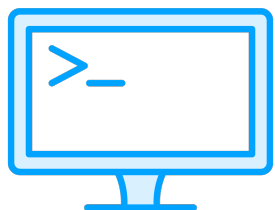
Password attacks



Security group modifications



Windows service modifications



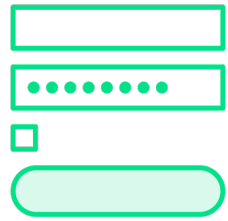
Command line execution



Malicious framework attacks



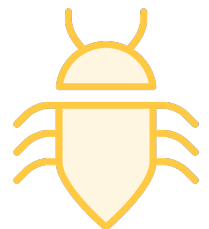
Example Anomalous Events



Password spraying via failed logon (Event IDs 4625, 4648 and 4771)



PowerShell "Net.WebClient" Downloadstring (Event ID 4688)



Suspicious service creation (Event IDs 4698 and 4699)



Example Anomalous Events

Get Password Spray Entries

```
Get-WinEvent -FilterHashtable @{ LogName = "Security"; ID = 4648 }
```

Get PowerShell Download String Entries

```
Get-WinEvent -FilterHashtable @{ LogName = "Security"; ID = 4688 }
```

Get Service Creation Entries

```
Get-WinEvent -FilterHashtable @{ LogName = "Security"; ID = 4698 }
```

Query Password Spray, PowerShell Download String, Service Creation Entries

```
$eventIDs = 4648,4688,4698
```

```
Get-WinEvent -FilterHashtable @{ LogName = "Security"; ID = $eventIDs }
```



Anomaly Classification

1

True positive (TP)

A true positive is a malicious action which is flagged

2

True negative (TN)

A true negative is a non-malicious action which is not flagged

3

False positive (FP)

A true negative is a non-malicious action that is flagged

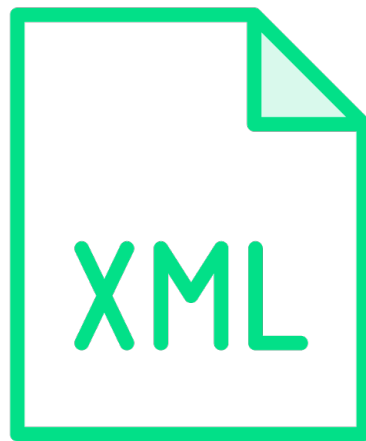
4

False negative (FN)

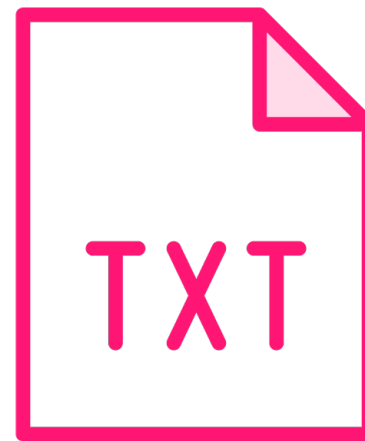
A false negative is a malicious action which is not flagged



Converting Event Logs



XML



TXT



CSV



Converting Event Logs

Convert to CSV

```
$log = "Security"
$path = "C:\Security-Export.csv"

Get-WinEvent -LogName $log | `
    Export-Csv $path
```

Convert to XML

```
$log = "Security"
$path = "C:\Security-Export.xml"

Get-WinEvent -LogName $log | `
    Export-Clixml $path
```



Demo

Analyzing Event Logs for Anomalies

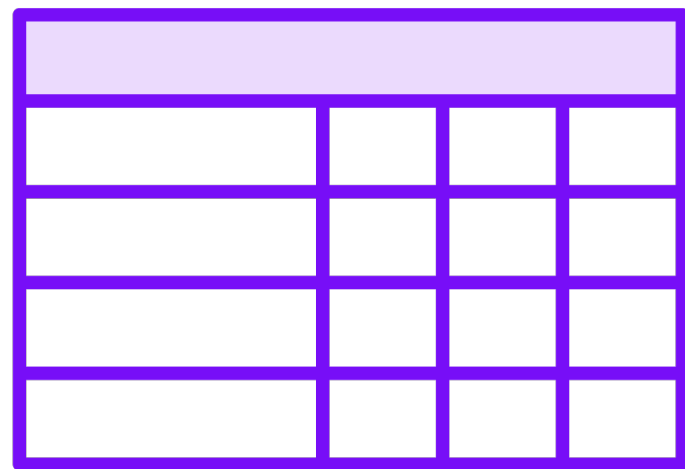
- Export event logs to EVTX
- Convert event logs to CSV, XML, and TXT
- Query events logs for anomalies



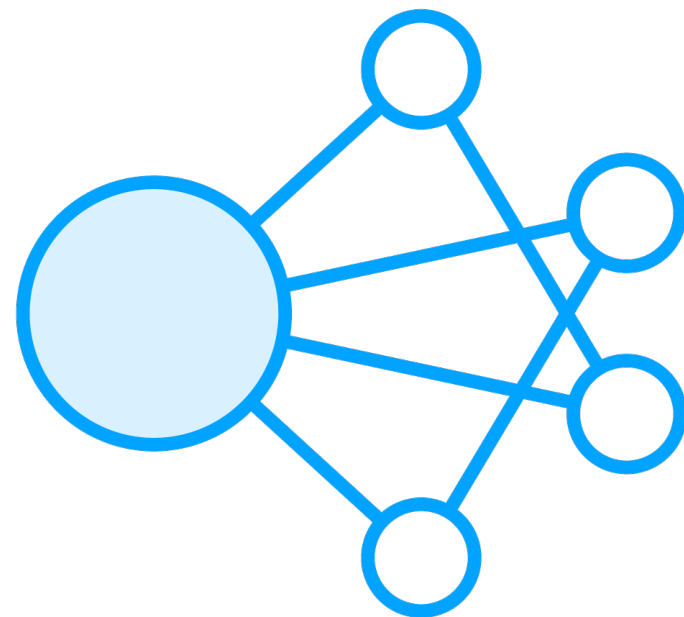
Writing Queries for Event Logs



Writing Queries for Event Logs



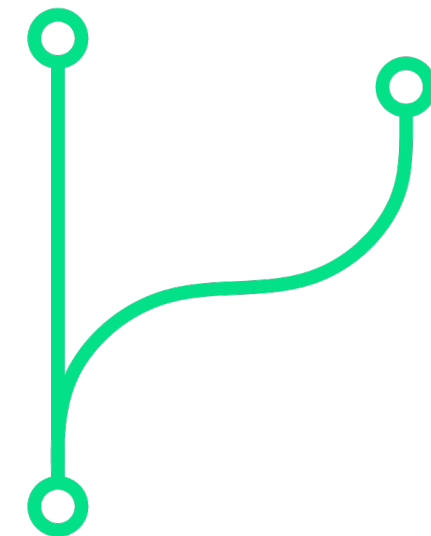
FilterHashtable



Where-Object



FilterXML



FilterXPath


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

Get-WinEvent -FilterHashtable @{ LogName = 'Application'; ID = 4688 }

$date = (Get-Date).AddDays(-1)
Get-WinEvent -FilterHashtable @{ LogName = 'Application'; StartTime = $date; Id = 4688 }

Get-WinEvent -FilterHashtable @{
    Logname = 'Application'
    ProviderName = 'Application Error'
    Data = 'calc.exe'
}
```

FilterHashtable

The FilterHashtable accepts a hash table as a filter to retrieve specific information from Windows event logs



Where-Object

Get Security Log Entries with an ID of 4688

```
Get-WinEvent -LogName Security | Where-Object { $_.ID -eq 4688 }
```

Get Security Log Entries Created within the Last 24 Hours

```
$date = (Get-Date) - (New-TimeSpan -Day 1)
```

```
Get-WinEvent -LogName Security | Where-Object { $_.TimeCreated -gt $date }
```

Get Security Log Entries with an ID of 4688 Created within the Last 30 Days

```
$eventIDs = 4648,4688,4698
```

```
$date = (Get-Date) - (New-TimeSpan -Day 30)
```

```
Get-WinEvent -LogName Security | `
    Where-Object { `
        $_.TimeCreated -gt $date -and $_.EventID -in $eventIDs }
```



FilterXML

Get Security Events by ID 4688

```
$query = @'  
  <QueryList>  
    <Query Id="0" Path="Security">  
      <Select Path="Security">  
        *[System[(EventID='4688')]]  
      </Select>  
    </Query>  
  </QueryList>  
'@  
Get-WinEvent -FilterXML $query
```



FilterXPath

```
# Get Security Events by ID 4688
```

```
$xpath = "*[System[(EventID = '4688')]]"
```

```
Get-WinEvent -LogName "Security" -FilterXPath $XPath
```

```
# Get Security Events Created within the Last 24 Hours
```

```
$xpath = '*[System[EventID = '4688' and TimeCreated[timediff(@SystemTime) <= 86400000]]]'
```

```
Get-WinEvent -LogName "Security" -FilterXPath $XPath
```



Demo

Writing Queries for Event Logs

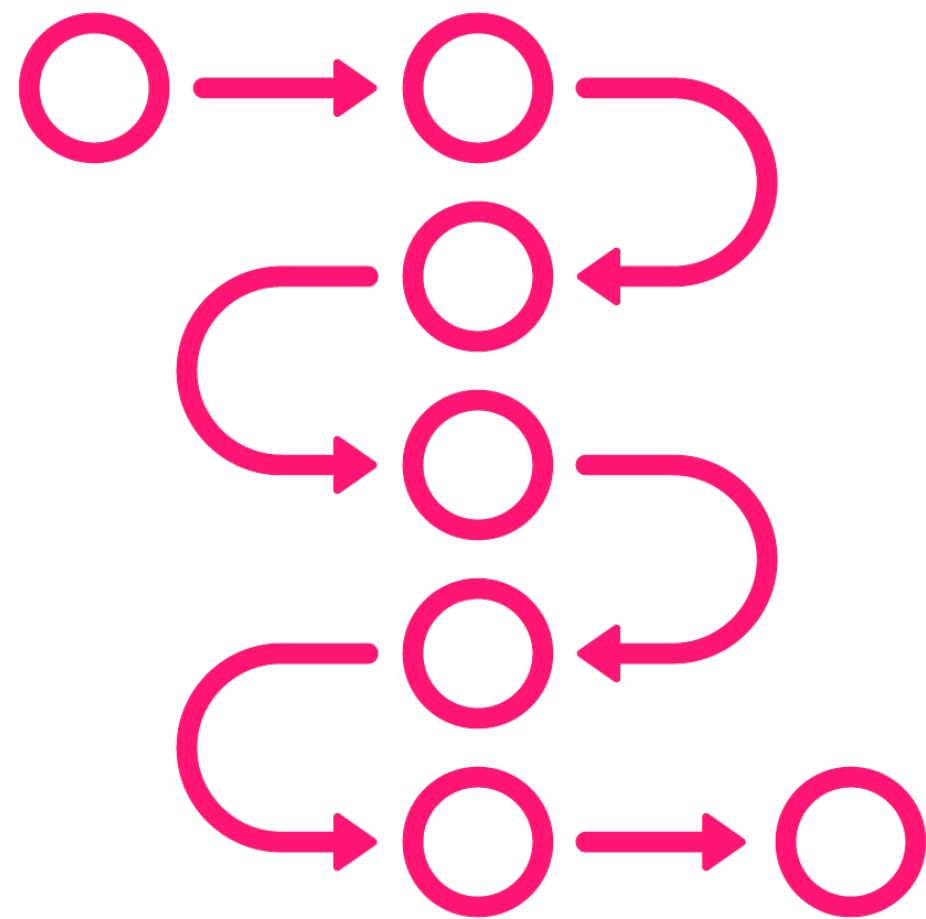
- Write queries using a FilterHashtable
- Filter using the Where-Object
- Use the FilterXML and FilterXPath for querying
- Write queries searching for specific values



Importing Event Log Entries into a Database



Importing Event Log Entries into a Database Tasks



Export event log entries to EVTX format

Create a database and table

Import each exported event log into the database table



Demo

Import Event Log Entries into a Database
Query Event Log Data within the Database



Summary

Goal: Analyze Collected Data for Anomalies

- Exported log entries as *.evtx files
- Analyzed log entries using standard and complex queries
- Imported event log entries into a database using PowerShell
- Queried event log data within the database using SQL and PowerShell

