Global Information Assurance Certification Paper

A Qradar Log Source Extension Walkthrough and Case Study

GIAC (GCIH) Gold Certification

Author: Michael Stanton, corenor@gmail.com
Advisor: Richard Carbone

Accepted:

Abstract

One of the first steps that any organization must take in order to reduce risk and improve overall security is to understand the activity that is taking place on their networks and computer systems. These activities are recorded in the log files these systems produce. As these systems have grown in speed and complexity, understanding this activity has become increasingly difficult. For this primary reason, the class of information technology solutions referred to as security information and event management or "SIEM" has increased in popularity in step with this trend. SIEM solutions continue to evolve as they are deployed within commercial enterprises, the public sector and academia. A leader in this field, IBM Qradar, is consistently given high marks for ease of configuration (Gartner 2014). Unfortunately, the availability of documentation for one of the most critical support aspects of this solution, building custom log source extensions, is quite limited. The vendor, IBM, provides a technical reference but illustrated or guided primers with examples are almost nonexistent. This paper will introduce SIEM, walk through the log source extension creation process and discuss how it complements the overall configuration process for the solution.

1. Introduction

The acronym SIEM refers to "Security Information and Event Management". Due to the many and varied functions provided, a concise definition is illusive. Some consistently offered functions across all SIEM platforms include log and threat correlation, incident management and reporting (Swift 2006).

Additional functions provided may also include active response, IT regulatory compliance, endpoint security, vulnerability assessment and advanced log search (Miller, Harris, Harper, VanDyke, Blask 2011).

Miller's definition leads to the conclusion that this class of multi-function system has evolved from a group of discrete software tools that each contribute to a unified security response platform (Miller 2011). Miller goes on to suggest that this is a logical evolution from such things as log aggregation platforms. As the solution matured, developers added correlation rules to assist with sifting through large amounts of disparate data. Ongoing improvements strive to minimize false positives by including more information such as threat and reputation feeds and host vulnerability data.

SIEM solutions offer the promise of solving or assisting with the solution of many cyber security challenges. For example, they can act as a repository for log data from other devices. They can also generate alerts when important security incidents are occurring. Additionally, they can provide workflows for tackling security incidents provide compliance or other reporting capabilities or assist with firewall management and log collection.

While SIEM offers many potential advantages, they are generally very expensive. If the cost of a security solution exceeds the benefit of avoiding potential losses then it may not be suitable for a particular organization. This would likely be the case for small organizations. Furthermore, the value will not be recognized if care and the required resources are not provided to configure and support the device as well as to monitor the logs and alerts provided.

2. Qradar Background

The Qradar SIEM was first introduced in 2001 (Miller 2011). This solution has consistently received high marks from Gartner who reviews and ranks them each year.

Michael Stanton, corenor@gmail.com

An alternative 2008 review from Network World Magazine noted the need for improvement across all tested platforms but concluded that Qradar was the overall leader (Shipley 2008). Shipley points out that "The products from Q1 Labs and eIQ supported the widest assortment of security devices and platforms out of the box" (Shipley 2008).

Over the last five years the solution has matured by improving support for vulnerability management and by better integrating rules that correlate flow data and log data. In addition, Qradar has positioned their product to support larger deployments by allowing specific components to be located on individual hosts rather than a single "all-in-one" console. Under the guidance of IBM, the product is also offered as either software only or as a virtual appliance (IBM 2014).

Additionally, Qradar has improved its ranking for each of the past four years. Gartner's 2014 ranking places Qradar ahead of all other solutions including the thirteen they included in their magic quadrant rankings. The Gartner "Magic Quadrant" compares solutions in two major categories, completeness of vision, and ability to execute (Gartner 2014).

One of the positive differentiators that Gartner highlights is the straightforward process for configuration (Nicolette 2014). If there is no "out of the box" support, then the process is more involved. However, while creating a log source extension is not very complex, it is poorly documented, as there are no "wizards" or other tools to help with this process. Regrettably, if you do dive in to adding a log source and it is not working properly, the console provides no feedback to assist with troubleshooting.

2.1 Major Components

This section will examine the major moving parts in Qradar in order to highlight the importance of properly indexed and cataloged event data.

2.1.1 Logs

Logs from various systems within the enterprise are one of two key information types that feed Qradar. This information source feeds the log correlation part of the overall solution. Most enterprise systems support the ability to configure a remote log destination using the Syslog protocol (Gerhards 2009). Some important log sources to include here are network, host intrusion detection and prevention systems, firewalls, authentication sources, and various operating systems and applications (Log Sources

Users Guide 2013). Our case study will be to configure Qradar to accept and accurately correlate log data from Ipswitch WS-FTP, a server that supports file hosting using FTP, SFTP, and SSH (FTP Server Software - WS FTP Server 2014).

This log source was selected for two reasons. A search of the IBM support site as well as broader searching of public information sources yielded no existing log source extension for this product. Therefore, this document, which provides a systematic procedure, may benefit other organizations needing to incorporate these logs with their SIEM deployment. In addition, the logs from this FTP server are complex enough to require the use of many different techniques. The final log source extension would offer guidance for many other log sources by illustrating the varied methods for creating these extension documents.

2.1.2 Flows

The second key information type that Qradar utilizes is flow data. Flow originally referred to the basic router accounting data that could be enabled on Cisco devices. The router could be configured to store statistics about the traffic that traversed the device. The Cisco website provides the following description of the protocol they created:

> "Cisco IOS NetFlow efficiently provides a key set of services for IP applications, including network traffic accounting, usage-based network billing, network planning, security, Denial of Service monitoring capabilities, and network monitoring. NetFlow provides valuable information about network users and applications, peak usage times, and traffic routing. Cisco invented NetFlow and is the leader in IP traffic flow technology" (Cisco 2014).

Request for comment (RFC) 3954 "Cisco Systems NetFlow Services Export Version 9" defines a flow as:

> "… a set of IP packets passing an Observation Point in the network during a certain time interval. All packets that belong to a particular Flow have a set of common properties derived from the data contained in the packet and from the packet treatment at the Observation Point" (Claise 2004).

There are now many different flow types. Competing network equipment providers created these protocol variations. Some variations include *sflow*, *jflow* and *qflow*; Qradar also supports these variations. If enabled, this information summarizes all

of the various flows that are transiting a particular router or other inspection device. Information that is captured includes IP layer information such as source and destination address, port information, bytes transferred, and flow duration. Flow logs also include useful information about the length of a connection and how much data was sent in either direction. As this type of information gathering has matured, it has expanded to include more details about application traffic. Additionally, the overall mechanism for collecting and sending this data has become more efficient as Qradar can also gather flow information by directly monitoring the network using a network tap or span port.

Flow data serves a useful function on a SIEM by mapping the flow of information in the environment. The other major source of traffic flow data is from firewall logs.

2.1.3 Assets and Vulnerabilities

A properly deployed SIEM platform collects large amounts of information relating to threat activity. Early correlation rules would process an exploit attempt and prioritize this as a critical alert. While exploits attempts, recon, brute force attacks, and other threat activity are all noteworthy when they are observed, if the target of the attack is not vulnerable then the priority of these events should be lowered. For example, an intrusion detection alert for a Solaris attack could automatically be downgraded in severity if the target was a Windows host because the attack would not succeed.

This asset and vulnerability data serves other purposes as well. For example, an asset repository is useful by itself. The list can be browsed and annotated. In addition, reports can be run against the list of assets. Additionally, an asset list can have a different aging setup than the raw log data. This way, even if the system has purged old attack data for a particular threat actor, the asset record can be updated to reflect that this host has been associated with prior incidents. This can help to discover the "low and slow" type of attacks. Moreover, it may be possible to detect an attacker by searching for new assets that have appeared inside the environment when none were deployed by the organization.

2.1.4 Building Blocks, Rules, and Offenses

As discussed, the Qradar engine receives log style events and flow data. The information contained is then analyzed against a set of rules. The rules are logical operators based on various match conditions used to determine if an "Offense" has

occurred. For a perfectly tuned Qradar SIEM an offense will represent a true security incident that would require investigation or activation of the incident response plan. Finally, building blocks are container objects that can be referenced in rules.

3. Logs In-Depth

3.1 Obtaining Logs

As the name implies, a log source extension is associated with a log source. Before a log source extension can be implemented the various logs should obtained. Sometimes, the vendor will provide a guide describing the types of logs their solution produces. If a guide is not available, it may be necessary to capture the logs using some other mechanism. One option is to enable logging and then to send events to an existing log server or to capture the logs directly using a packet capture tool such as *tcpdump*. Another option is to send the logs to Qradar and capture them using the universal device support module. Having a sample of each type of log that can be produced is essential for creating a comprehensive log source extension and for mapping all of the possible types of events.

3.2 Reviewing Logs for Usability

Once a representative sample of logs has been collected, the next step is to examine them to determine which fields can be mapped to existing events, correlated with existing fields, or provide additional useful information while analyzing the logs. However, the "event" field is the most important.

An event will map to a Qradar identifier or "QID". Once the appropriate QID has been determined for a particular log, supporting values like username, source/destination IP, source/destination port or MAC address are also desirable. This process will be discussed later on in the case study.

3.3 Log source formats

Qradar supports many different options for loading log data. Typical methods include various Syslog flavors, SNMP, and file scraping. They also support specific protocols that vendors may require including certificate based SSL connections, ODBC and the Open Platform for Security (OPSEC).

3.3.1 Useful Fields within a Log Entry

A common support task for any SIEM deployment will be to locate and incorporate useful log sources. A log source should first be evaluated to determine if the data that it provides could help with discovering and responding to security incidents. Currently, the Qradar documentation lists over 250 different supported devices or protocols (DSM Configuration Guide 2013).

3.4 Inclusion Methods from Easiest to Most Difficult

There are four common methods for integrating log data in Qradar. These methods vary in difficulty. Each has use cases that make it an appropriate choice and are examined in the subsequent sections.

3.4.1 Generic Inclusion

By far the easiest way to collect unsupported log data is to send it to the device and associate the logs with a generic or "universal" device support module or "DSM". This may be appropriate if the only requirement is to store the data but there is no value from a security perspective. Perhaps there is little fear that a host will be compromised but the logs from that host should be retained for troubleshooting or to satisfy compliance requirements.

3.4.2 Extract Property

If a log source such as the host described above is being collected using the universal "DSM" there may be a few fields within that raw log record that have investigative value. For example, the log may have a username and that username might match with other log sources to help analyze security incidents. The process for extracting a specific property is to locate a sample log within the Qradar console and choose "Extract Property" from the menu. The resulting dialog will ask for a name for the property and give a guided dialog for creating a regular expression to locate the property within the log sample. One glaring problem with this method is that it is not possible to map an extracted property to one of the system provided index fields like "Source IP" or "Username". Instead, a custom field must be created for the property. The custom field can be searched but it will not be referenced in any correlation rules.

3.4.3 Log Source Extension

This option will be discussed in greater detail later. A log source extension or "LSX" is an XML formatted file that defines how the elements of a particular log are labeled.

This method provides some significant benefits over extracting individual properties. Firstly, a log source extension can be moved around. If you have ten consoles that operate independently, it is much easier to upload the log source extension document to each console rather than walking through the property extraction procedure for each property on each device. This log source extension could also be shared with others outside the organization. Maybe the owner of the log source would find the extension useful! In addition, unlike extract property, it is possible to match fields in a log source extension against the primary indexed fields in Qradar, which is critical so that the new log source can be included in searches and various rules.

3.4.4 Device Support Module (A.K.A Call the Vendor)

For common security related logs Qradar ships with a large number of DSMs. A DSM is a bundle of scripts that include the elements of a log source extension as well as configuration parameters for mapping events received to the appropriate rules. The DSM may also update building block groups and other lists in order to integrate with existing rules. Unfortunately, IBM does not support user created DSMs and will void the warranty or at the very least necessitate rebuilding or restoring the SIEM if something breaks.

4. Log Source Extension (Case Study)

As discussed, IBM provides a log source extension XML template. The template can be tailored to support new log sources. Once the log source extension is created, only a few steps are necessary to upload the file, add a log source and associate it with the extension template.

4.1 Step 1: Exporting Logs for Evaluation

Once a potential log source has been identified, the first objective is to export some of the logs in order to determine what fields are available. Collect enough of the logs to account for as many variations as possible. The log source use for the case study was unusual in that there was no uniform format. The application "WS_FTP" functions

as a traditional FTP server but can also serve encrypted SFTP as well as SSH. Each of these logs then contained various event types including establishing or tearing down connections (ws_ftp 2014).

Below are some examples of the variations that are produced from this log source:

```
<13>Aug  3 14:32:58 SERVER FTP: Connection closed <SessionID=58616793,
Listener=198.147.12.43:21, Client=172.25.254.27:27515><Command=start>
<11>Aug  3 14:32:44 SERVER SSH: No User. Possible reasons: Invalid
username, invalid license, error while accessing user database
<SessionID=29634951, Listener=198.147.12.43:22, Client=172.25.254.27:8077,
User=infosec>
<14>Aug  3 14:28:17 SERVER FTP: SendResponse: An existing connection was
forcibly closed by the remote host <SessionID=19346180,
Listener=198.147.12.43:21, Client=172.25.254.27:19971><Command=QUIT,
Error=10054>
        <13>Aug  3 12:10:00 SERVER SSH: Completed password Authentication.
User logged in <Host=usftp.discovery.com, SessionID=41272593,
Listener=198.147.12.43:22, Client=50.58.39.77:63163, User=ussidecoder>
        4>Aug  3 12:10:01 SERVER SSH: Client closed connection
<Host=usftp.discovery.com, SessionID=41272593, Listener=198.147.12.43:22,
Client=50.58.39.77:63163, User=ussidecoder>
        <13>Aug  3 13:01:36 SERVER FTP: <Host=usftp.discovery.com,
SessionID=62734836, Listener=198.147.12.43:21, Client=172.30.31.81:11632,
User=netcaptioning><Command=STOR,
Parameters=144356.040.01.285A.txt><Filename=\\us\shared\usftp\private\netca
ptioning\Offline Files\Hub\144356.040.01.285A.txt, FileSize=51648 bytes,
TransferTime=1078 ms>
```

The log samples have some consistency but there are some challenging deviations. Certain value pairs are present in most logs like “Client=” while others are context specific such as “Filename=”. In addition, in some log entries the text that follows the session type “FTP:”, “SSH:” or “SFTP:” is a good value to use for the event

name. In other cases the value that follows "Command=" is more useful. In some cases, the value that follows the session type also includes specific unique information. For example, it may reference a filename. Therefore, if that were included verbatim as the event name, then it would always be a unique value, which is particularly not useful. We will discuss how some of these challenges are addressed when looking at the case study log source extension.

From the log export exercise, we have examined the output and come up with the following table of useful information:

Attribute	Mapping Information
Event Name	The information following the session start information like "SSH:" could be used, also what follows "Command="
Source IP	The value is delivered as "Client=" within the log source
Source Port	The value follows the source IP separated by a colon.
Destination IP	The value is delivered at "Listener=" within the log source
Destination Port	The value follows the destination IP separated by a colon.
Username	The value is delivered at "User=" within the log source
Hostname	The value is delivered at "Host=" within the log source

Table 1: WS_FTP Useful Field Identifiers for Raw Logs

4.2 Step 2: Creating Regular Expressions to Capture Values

Once the useful information has been identified, the next step is to figure out what regular expressions would locate information within the log. Qradar uses Java style regular expressions. Java regex supports "capture groups" which means it is possible to match a string of data within the log source and then divide it up into pieces; a particular value could be one of the various pieces. The different capture groups within the regular expression are separated by parentheses "( )" and numbered sequentially starting at one.

For our case study, the following regular expressions work well and consistently to find the information necessary within the log data:

Attribute	**Mapping Information, the parentheses characters are marked in red to show where the value is matched.**
Event Name	FTP:\s(.\w+\s\w+) The character string “FTP:” followed by a space, then the next two words. SSH:\s(.\w+\s\w+) “The character string “SSH:” followed by a space, then the next two words. SFTP:\s(.\w+\s\w+) channel :\s(.\w+\s\w+) COM_API:\s(.\w+\s\w+) Likewise pattern matching for above anchor strings. <Command=([a-zA-Z]+) The letter string following character string “<Command=”
Source IP	Client=([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+):
Source Port	Client=[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+:([0-9]+)
Destination IP	Listener=([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)
Destination Port	Listener=[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+:([0-9]+)
Username	User=([a-zA-Z]+)
Hostname	\<Host=([a-zA-Z]+\.[a-zA-]+\.[a-zA-Z]+[\.*a-zA-Z]*)

Table 2: Regular Expression Pseudocode

In the above example under Event Name, the value “FTP:” is used as an anchor in the log data. The regular expression “\s(.\w+\s\w+)” expects a space after the “FTP:” and then will match the next two words. Because only the two words are in the parenthesis, they will represent the event name. This example only has one parenthesis so this would only have one “capture” group.

4.3 Step 3: Applying Values to the Template

Appendix A shows the unaltered template provided by IBM. Below is the updated template incorporating the useful fields derived from analysis of the ws_ftp log data export:

```
<?xml version="1.0" encoding="UTF-8"?>
<device-extension xmlns="event_parsing/device_extension">
    <!-- Do not remove the "allEventNames" value -->
    <pattern id="allEventNames" xmlns=""><![CDATA[(.*)]]></pattern>

    <!-- Everything below this line can be modified -->
    <pattern id="EventName0" xmlns=""><![CDATA[FTP:\s(.\w+\s\w+)]]></pattern>
    <pattern id="EventName1" xmlns=""><![CDATA[SSH:\s(.\w+\s\w+)]]></pattern>
    <pattern id="EventName2" xmlns=""><![CDATA[SFTP:\s(.\w+\s\w+)]]></pattern>
    <pattern id="EventName3" xmlns=""><![CDATA[channel:\s(.\w+\s\w+)]]></pattern>
    <pattern id="EventName4" xmlns=""><![CDATA[COM_API:\s(.\w+\s\w+)]]></pattern>
    <pattern id="EventName5" xmlns=""><![CDATA[<Command=([a-zA-Z]+)]]></pattern>
    <pattern id="EventName6" xmlns=""><![CDATA[\s(.*)<]]></pattern>

    <!-- Additional useful fields to match -->
    <pattern id="SourceIp" xmlns="">
        <![CDATA[Client=([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+):]]>
        </pattern>
    <pattern id="SourcePort" xmlns="">
        <![CDATA[Client=[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+:([0-9]+)]]>
        </pattern>
    <pattern id="DestinationIp" xmlns="">
        <![CDATA[Listener=([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)]]>
        </pattern>
    <pattern id="DestinationPort" xmlns="">
        <![CDATA[Listener=[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+:([0-9]+)]]>
        </pattern>
    <pattern id="UserName" xmlns="">
        <![CDATA[User=([a-zA-Z]+)]]>
        </pattern>
    <pattern id="HostName" xmlns="">
        <![CDATA[\<Host=([a-zA-Z]+\.[a-zA-]+\.[a-zA-Z]+[\.*a-zA-Z]*)]]>
        </pattern>

<match-group order="1" description="Log Source Extension" xmlns="">

    <!-- Below determines matching order for the 7 event names above -->
<matcher field="EventName" order="1" pattern-id="EventName0" capture-group="1"
        enable-substitutions="false"/>
<matcher field="EventName" order="2" pattern-id="EventName1" capture-group="1"
        enable-substitutions="false"/>
<matcher field="EventName" order="3" pattern-id="EventName2" capture-group="1"
        enable-substitutions="false"/>
<matcher field="EventName" order="4" pattern-id="EventName3" capture-group="1"
        enable-substitutions="false"/>
<matcher field="EventName" order="5" pattern-id="EventName4" capture-group="1"
         enable-substitutions="false"/>
<matcher field="EventName" order="6" pattern-id="EventName5" capture-group="1"
        enable-substitutions="false"/>
<matcher field="EventName" order="7" pattern-id="EventName6" capture-group="1"
        enable-substitutions="false"/>

    <!-- These relate to the additional useful fields above -->
<matcher field="SourceIp" order="1" pattern-id="SourceIp" capture-group="1" />
<matcher field="SourcePort" order="1" pattern-id="SourcePort" capture-group="1" />
<matcher field="DestinationIp" order="1" pattern-id="DestinationIp" capture-group="1" />
<matcher field="DestinationPort" order="1" pattern-id="DestinationPort" capture-group="1" />
<matcher field="UserName" order="1" pattern-id="UserName" capture-group="1" />
<matcher field="HostName" order="1" pattern-id="HostName" capture-group="1" />

    <!--- From Template --->
        <event-match-multiple pattern-id="allEventNames" capture-group-index="1"
        device-event-category="unknown" send-identity="OverrideAndNeverSend" />
    </match-group>
</device-extension>
```

Figure 1: Case Study LSX

Michael Stanton, corenor@gmail.com

As discussed, one challenge for our log source is that there are various places within the logs that could potentially match against the critical "event name". The LSX provides for this by letting you try to match as many times as you wish. If the first attempt fails then proceed to the next one. We have six different attempts to match an event name:

```
<pattern id="EventName0" xmlns=""><![CDATA[FTP:\s(.\w+\s\w+)]]></pattern>
<pattern id="EventName1" xmlns=""><![CDATA[SSH:\s(.\w+\s\w+)]]></pattern>
<pattern id="EventName2" xmlns=""><![CDATA[SFTP:\s(.\w+\s\w+)]]></pattern>
<pattern id="EventName3" xmlns=""><![CDATA[channel:\s(.\w+\s\w+)]]></pattern>
<pattern id="EventName4" xmlns=""><![CDATA[COM_API:\s(.\w+\s\w+)]]></pattern>
<pattern id="EventName5" xmlns=""><![CDATA[<Command=([a-zA-Z]+)]]></pattern>
<pattern id="EventName6" xmlns=""><![CDATA[\s(.*)<]]></pattern>
```

Figure 2: Pattern Match for EventName from Case Study LSX

The event names are each given a unique name. In this case, we have numbered them sequentially. The "pattern id" value will be referenced further down in the LSX file.

4.2.1 Using Multiple Capture Groups with a Single Java Regular Expression

Each reference in the "matcher field" includes a "capture-group" reference. Below, an excerpt shows the matcher fields each referencing, in this case, capture group 1:

```
    <!-- Below determines matching order for the 7 event names above -->
<matcher field="EventName" order="1" pattern-id="EventName0" capture-group="1"
        enable-substitutions="false"/>
<matcher field="EventName" order="2" pattern-id="EventName1" capture-group="1"
        enable-substitutions="false"/>
<matcher field="EventName" order="3" pattern-id="EventName2" capture-group="1"
        enable-substitutions="false"/>
<matcher field="EventName" order="4" pattern-id="EventName3" capture-group="1"
        enable-substitutions="false"/>
<matcher field="EventName" order="5" pattern-id="EventName4" capture-group="1"
         enable-substitutions="false"/>
<matcher field="EventName" order="6" pattern-id="EventName5" capture-group="1"
        enable-substitutions="false"/>
<matcher field="EventName" order="7" pattern-id="EventName6" capture-group="1"
        enable-substitutions="false"/>
```

Figure 3: Matcher Field Assertions for Event Name

The matcher field above refers to the system field "EventName". In this case, the LSX attempts to match against "EventName" first by checking the value from the

"pattern id" in Figure 1 "….FTP:…" referred to as "EventName0". This continues for each of the event names following the order requested by the "order" operator.

Another way to set this up, depending on the log data, is to have a single "pattern_id" at the top and then match against different parts. In the case study example, if the last octet of the source IP address is useful to separately show, the matcher reference could look for that specific field.

In this example, consider the seventh capture group:

```
<matcher field="SourceIPLastOctet" order="1" pattern-id="SourceIp"
capture-group="7" />
```

This expression would match if we separated each octet in the pattern ID using parentheses. The seventh match group is represented in bold below:

```
<pattern id="SourceIp" xmlns=""><![CDATA[Client=([0-9]+)(\.)([0-
9]+)(\.)([0-9]+)(\.)([0-9]+):]]></pattern>
```

4.4 Step 4: Installing the Log Source Extension

The procedure for installing the log source extension is simple and instructions are available from the Qradar documentation. The first step is to upload the log source extension. There is a log source extension management GUI available within the Admin tab of Qradar. If the log source extension has the proper syntax, it will be accepted when uploaded to the console following the steps under the "add" action shown below.

Figure 4 illustrates the log source extension management popup window. This menu is accessed from the data sources section of the Admin tab.

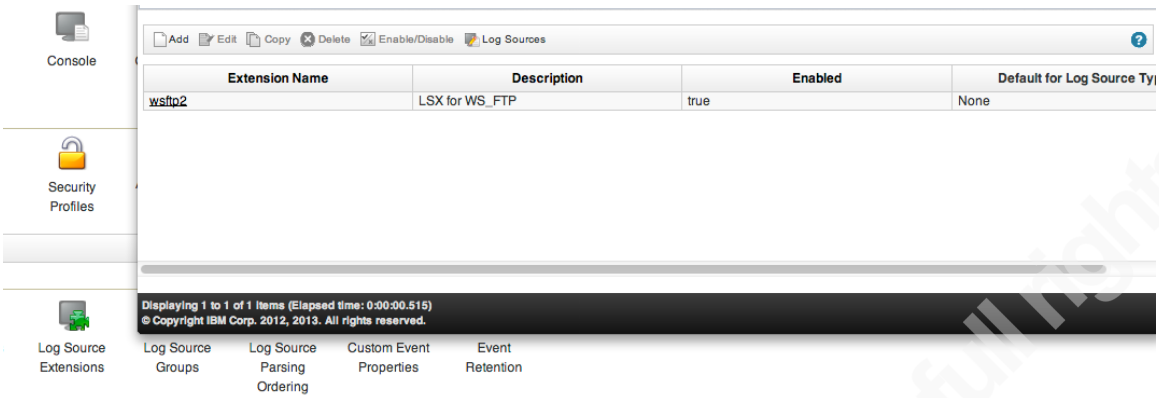


Figure 4: LSX Install Dialog

The next step is to create a generic log source using the "Universal DSM". The log source needs to match against the data received from the log source. The reference log source for our FTP server looks like Figure 5. For our source, "SERVER" would be replaced with the actual hostname that was included in the logs.

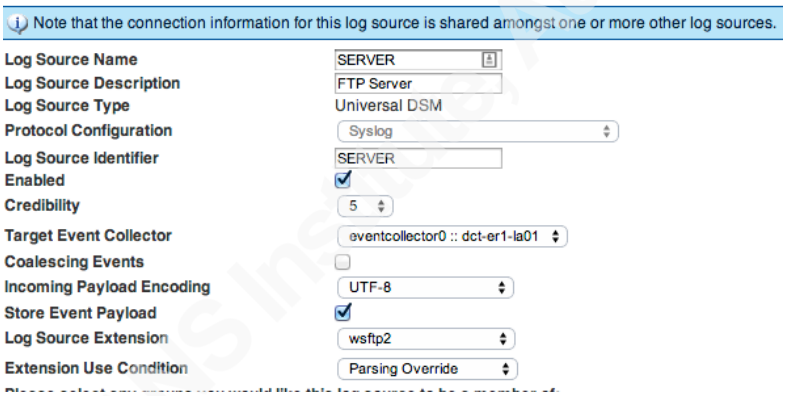


Figure 5: Qradar Log Source Properties Dialog

If the regular expression used to match against event names is working then the events should start appearing in the Qradar log window. At this point, the event name will show the "unknown" on the Qradar log viewer. Correctly matched fields such as source IP and Destination IP indicate proper functioning of the LSX. Figure 6 illustrates correctly parsed log data that requires further event mapping.

Figure 6: Qradar Log Viewer WS_FTP Case Study without Mapping

Michael Stanton, corenor@gmail.com

4.5 Step 5: Mapping Events

Unlike a device support module, a log source extension does not have any built in logic to help with event mapping. Each event shows "unknown", however, we identified a place within each log to search for some text that will give us information on how to map a particular event. As each unknown event is matched, it will start to show that new event name each time that type of event is seen.

For the raw log, consider:

```
<13>Aug  6 14:15:23 SERVER FTP: Connection established
<SessionID=31347380, Listener=100.100.100.100:21,
Client=200.200.200.200:51072><Command=start>
```

Our log source extension matched the two words after the anchor value "FTP:". This is the first assertion from the log source extension:

```
<pattern id="EventName0" xmlns=""><![CDATA[FTP:\s(.\w+\s\w+)]]></pattern>
```

There are many great resources for helping with regular expression creation. Qradar complies with the "Java" regex standards and the website "Tutorials Point" offers a good tutorial (tutorialspoint 2014).

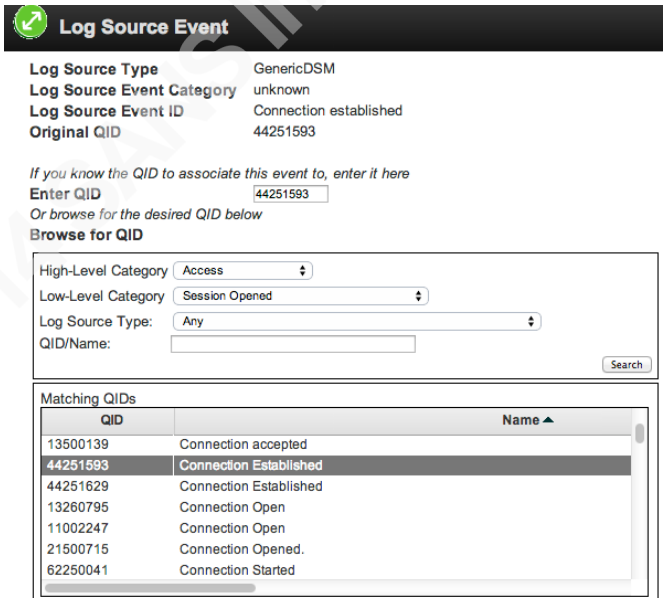
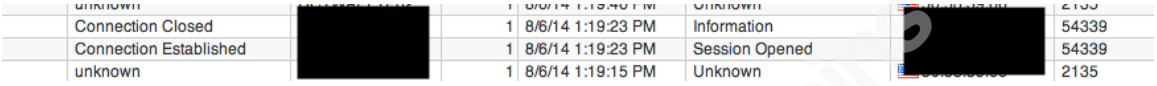


Figure 7: Qradar Event Mapping Dialog

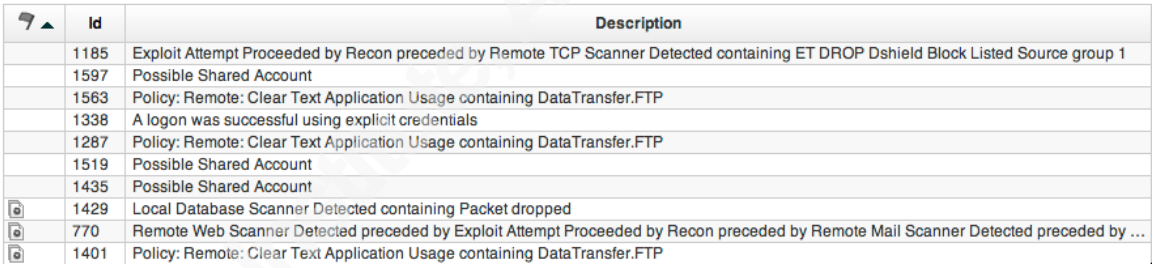
Figure 7 displays the dialog for mapping events. A button titled "Map Event" is available from the menu while viewing an event detail. This will give a pop up box showing the event name we are searching for in the log. There is a search dialog for selecting an available event name or "Qradar Identifier" or QID. I chose to match this with the QID 44251593. This maps the high-level category for the event as "Access", the low-level category as "Session Opened" and the event name as "Connection Established". Once this is in place, any new events of this type will look like Figure 8:



	Connection Closed	[redacted]	1	8/6/14 1:19:23 PM	Information	54339
	Connection Established		1	8/6/14 1:19:23 PM	Session Opened	54339
	unknown		1	8/6/14 1:19:15 PM	Unknown	2135

Figure 8: Qradar Log Viewer WS_FTP Case Study with Mapping

This data will also begin to populate any rules that reference this particular Qradar identifier. When these rules generate offenses, they will appear similar to those illustrated in Figure 9:



	Id	Description
	1185	Exploit Attempt Proceeded by Recon preceded by Remote TCP Scanner Detected containing ET DROP Dshield Block Listed Source group 1
	1597	Possible Shared Account
	1563	Policy: Remote: Clear Text Application Usage containing DataTransfer.FTP
	1338	A logon was successful using explicit credentials
	1287	Policy: Remote: Clear Text Application Usage containing DataTransfer.FTP
	1519	Possible Shared Account
	1435	Possible Shared Account
	1429	Local Database Scanner Detected containing Packet dropped
	770	Remote Web Scanner Detected preceded by Exploit Attempt Proceeded by Recon preceded by Remote Mail Scanner Detected preceded by ...
	1401	Policy: Remote: Clear Text Application Usage containing DataTransfer.FTP

Figure 9: Sample Offenses Resulting from Attacks against WS_FTP

5. Conclusions

Creating and using log source extensions is a key element of any successful Qradar implementation. This useful activity can turn a marginal SIEM deployment in to a successful one by including additional log sources that are not supported by the vendor but are critical data elements for discovering security incidents. Consider the case study. Prior to our log source extension implementation, we would never see SIEM security events resulting from attacks against our FTP server. While this procedure is not difficult, it is poorly documented and lacks sufficient guidance. Hopefully, this case study has provided some insights in to how a SIEM can be implemented within the enterprise to better discover and respond to security incidents. We have attempted to

highlight some of the key elements of a log source extension with the goal of providing insights for applying this process to other unknown log sources.

6. References

Cisco IOS NetFlow. (n.d.). Retrieved August 14, 2014.

Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, October 2004.

FTP Server Software - WS FTP Server | Ipswitch File Transfer. (n.d.). Retrieved August 30, 2014.

Magic Quadrant Research Methodology | Gartner Inc. (n.d.). Retrieved September 4, 2014.

Gerhards R., The Syslog Protocol - http://tools.ietf.org/html/rfc5424 2009. Supercedes - http://tools.ietf.org/html/rfc3164 2001.

IBM Log Sources User Guide. (2013, January 1). Retrieved July 14, 2014, from ftp://ftp.software.ibm.com/software/security/products/qradar/documents/71MR1/LogMgr/LogSources-71MR1.pdf

IBM Qradar DSM Configuration Guide. (2013, January 1). Retrieved July 30, 2014.

IBM Security QRadar SIEM All-in-One Virtual 3190 V7.2.2. (n.d.). Retrieved August 30, 2014.

Miller, Harris, Allen et. al, Security Information and Event Management (SIEM) Implementation. 2011

Nicolette, M., & Kavanagh, K. (2014, June). Magic Quadrant for Security Information and Event Management. Retrieved August 30, 2014.

Pinned topic Unofficial step by step Log Source/DSM Extension Guide. (2013, May 3). Retrieved August 30, 2014.

Shipley, G. (2008, June 30). SIEM tools come up short. Retrieved August 29, 2014.

Swift, D. (2006, December 26). A Practical Application of SIM/SEM/SIEM Automating Threat Identification. Retrieved August 29, 2014.

Tutorials Point - Simply Easy Learning. (n.d.). Retrieved August 29, 2014.

Michael Stanton, corenor@gmail.com

7. Appendix A Log Source Extension Template

```
<?xml version="1.0" encoding="UTF-8"?>
<!--

Author:             Your Name <your.email@q1labs.com>
Device Type:        Example SampleTron 5000 (FakeOS)
Device Version:     Fakeware 2.7.1
Protocol:           Syslog

Custom Property regular expressions for Event Viewer:
Sample-ID:                  \sPolicy\sID\:\s(.*?)\;
Sample-Group:               \sGroup\sName\:\s(.*?)\;

Common Regular Expressions:
IP Address:         \d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}
Port Number: \d{1,5}
MAC Address:(?:[0-9a-fA-F]{2}\:){5}[0-9a-fA-F]{2}
Protocol:           (tcp|udp|icmp|gre)
Device Time:  \w{3}\s\d{2}\s\d{2}:\d{2}:\d{2}
White Space:  \s
Tab:                \t
Match Anything:     .*?

-->
<device-extension xmlns="event_parsing/device_extension">
        <!-- Do not remove the "allEventNames" value -->
        <pattern id="allEventNames" xmlns=""><![CDATA[(.*)]]></pattern>
        <!-- Everything below this line can be modified -->
```

Michael Stanton, corenor@gmail.com

```
        <pattern id="EventName" xmlns=""><![CDATA[]]></pattern>
        <pattern id="EventCategory" xmlns=""><![CDATA[]]></pattern>
        <pattern id="SourceIp" xmlns=""><![CDATA[]]></pattern>
        <pattern id="SourcePort" xmlns=""><![CDATA[]]></pattern>
        <pattern id="SourceIpPreNAT" xmlns=""><![CDATA[]]></pattern>
        <pattern id="SourceIpPostNAT" xmlns=""><![CDATA[]]></pattern>
        <pattern id="SourceMAC" xmlns=""><![CDATA[]]></pattern>
        <pattern id="SourcePortPreNAT" xmlns=""><![CDATA[]]></pattern>
        <pattern id="SourcePortPostNAT" xmlns=""><![CDATA[]]></pattern>
        <pattern id="DestinationIp" xmlns=""><![CDATA[]]></pattern>
        <pattern id="DestinationPort" xmlns=""><![CDATA[]]></pattern>
        <pattern id="DestinationIpPreNAT" xmlns=""><![CDATA[]]></pattern>
        <pattern id="DestinationIpPostNAT" xmlns=""><![CDATA[]]></pattern>
        <pattern id="DestinationPortPreNAT" xmlns=""><![CDATA[]]></pattern>
        <pattern id="DestinationPortPostNAT" xmlns=""><![CDATA[]]></pattern>
        <pattern id="DestinationMAC" xmlns=""><![CDATA[]]></pattern>
        <pattern id="DeviceTime" xmlns=""><![CDATA[]]></pattern>
        <pattern id="Protocol" case-insensitive="true"
xmlns=""><![CDATA[]]></pattern>
        <pattern id="UserName" xmlns=""><![CDATA[]]></pattern>
        <pattern id="HostName" xmlns=""><![CDATA[]]></pattern>
        <pattern id="GroupName" xmlns=""><![CDATA[]]></pattern>
        <pattern id="NetBIOSName" xmlns=""><![CDATA[]]></pattern>
        <match-group order="1" description="Log Source Extension" xmlns="">
                <matcher field="EventName" order="1" pattern-id="EventName"
capture-group="1" enable-substitutions="false"/>
                <matcher field="EventCategory" order="1" pattern-
id="EventCategory" capture-group="1"/>
                <matcher field="SourceIp" order="1" pattern-id="SourceIp" capture-
group="1" />
```

```
                <matcher field="SourcePort" order="1" pattern-id="SourcePort"
capture-group="1" />
                <matcher field="SourceIpPreNAT" order="1" pattern-
id="SourceIpPreNAT" capture-group="1" />
                <matcher field="SourceIpPostNAT" order="1" pattern-
id="SourceIpPostNAT" capture-group="1" />
                <matcher field="SourceMAC" order="1" pattern-id="SourceMAC"
capture-group="1" />
                <matcher field="SourcePortPreNAT" order="1" pattern-
id="SourcePortPreNAT" capture-group="1" />
                <matcher field="SourcePortPostNAT" order="1" pattern-
id="SourcePortPostNAT" capture-group="1" />
                <matcher field="DestinationIp" order="1" pattern-id="DestinationIp"
capture-group="1" />
                <matcher field="DestinationPort" order="1" pattern-
id="DestinationPort" capture-group="1" />
                <matcher field="DestinationIpPreNAT" order="1" pattern-
id="DestinationIpPreNAT" capture-group="1" />
                <matcher field="DestinationIpPostNAT" order="1" pattern-
id="DestinationIpPostNAT" capture-group="1" />
                <matcher field="DestinationPortPreNAT" order="1" pattern-
id="DestinationPortPreNAT" capture-group="1" />
                <matcher field="DestinationPortPostNAT" order="1" pattern-
id="DestinationPortPostNAT" capture-group="1" />
                <matcher field="DestinationMAC" order="1" pattern-
id="DestinationMAC" capture-group="1" />
                <matcher field="DeviceTime" order="1" pattern-id="DeviceTime"
capture-group="1" />
                <matcher field="Protocol" order="1" pattern-id="Protocol" capture-
group="1" />
```

```
                <matcher field="UserName" order="1" pattern-id="UserName"
capture-group="1" />
                <matcher field="HostName" order="1" pattern-id="HostName"
capture-group="1" />
                <matcher field="GroupName" order="1" pattern-id="GroupName"
capture-group="1" />
                <matcher field="NetBIOSName" order="1" pattern-
id="NetBIOSName" capture-group="1" />
                <event-match-multiple pattern-id="allEventNames" capture-group-
index="1" device-event-category="unknown" send-
identity="OverrideAndAlwaysSend" />
        </match-group>
</device-extension>
```