Splunk and Logging with the Oracle ZFS Storage Appliance
Table of Contents
Introduction .............................................................................................................. 2
Overview of the Example System ............................................................................. 3
  Why Use a Forwarder? ......................................................................................... 3
Installing the Universal Forwarder ............................................................................ 6
Collecting Logs from the Oracle ZFS Storage Appliance ........................................ 7
  The `get_appliance_logs.py` Script ................................................................ 7
  Configuring the Forwarders to Parse Timestamps ................................................ 8
  Defining a Monitor .............................................................................................. 8
Using the Oracle ZFS Storage Appliance Log Data ............................................... 10
  Example: Charting Logins .................................................................................... 12
Conclusion .................................................................................................................. 14
References ................................................................................................................. 15
Appendix A: Python code for `get_appliance_logs.py` ........................................ 16
Appendix B: Adding a Non-Privileged User to the Oracle ZFS Storage Appliance ........................................................................................................... 20
Appendix C: Enabling the RESTful Interface ........................................................ 21
Introduction

The Operational Intelligence platform Splunk Enterprise is used by many organizations to gather machine data from many sources. The collected data is then used to monitor the end-to-end infrastructure to help avoid service degradation and to troubleshoot problems such as performance bottlenecks.

This white paper provides information on the fundamentals of collecting pertinent logging data from the Oracle ZFS Storage Appliance, basics on a Splunk infrastructure that allows distributing the load of collecting the data, and configuring Splunk Enterprise to index that data. The paper is intended for an audience already familiar with the Splunk Enterprise command line interface (CLI) and the Oracle ZFS Storage Appliance.
Overview of the Example System

The Splunk Enterprise infrastructure is flexible in how it collects its machine data. The heart of the Splunk infrastructure is the Indexer node. The Indexer node transforms the raw data into events and places the result into an index. It also searches the index in response to search requests.¹

In smaller deployments, the Indexer may also perform the other fundamental Splunk operations – data input and search management – but in larger deployments, these operations may be offloaded to other machines. A machine that takes on the data input operation is called a Forwarder. A Forwarder node runs a streamlined, dedicated version of Splunk Enterprise that contains only the essential components needed to forward data.²

Why Use a Forwarder?

In the Splunk infrastructure, the Indexer nodes can be quite heavily loaded, especially in a large environment. The tasks of data input can accumulate quickly, leading to poor performance at both server and network chokepoints. By using Forwarders, the machine resources can be distributed across a number of small machines or virtual machines (VMs), and the generated network traffic is more easily balanced.

In an environment where the monitored systems are spread across networks, Forwarders automatically queue the data if connection to the Indexers is lost, whereas if data is streaming directly to an Indexer, events can be lost during the network down time.

Another benefit to the use of Forwarders is the ability to group data from related machines, allowing faster access to related events on the Indexers.

¹ [http://docs.splunk.com/Splexicon:Indexer](http://docs.splunk.com/Splexicon:Indexer)
² [http://docs.splunk.com/Splexicon:Forwarder](http://docs.splunk.com/Splexicon:Forwarder)
Figure 1 illustrates an example of a distributed Splunk infrastructure. It covers three distinct islands of functionality, each on a separate subnet with limited routing between them. Each island has one or more Oracle ZFS Storage Appliances containing data pulled by a Forwarder which processes the raw data and pushes it to an Indexer.

This white paper details how the logs from the Oracle ZFS Storage Appliance in Development are collected by a Forwarder and pushed to an Indexer. This example assumes an environment with an Indexer node with Splunk Enterprise 6.2 installed and a server running Oracle Linux 6 with the Splunk Universal Forwarder installed.

To access the RESTful interface on the Oracle ZFS Storage Appliance, it must first be enabled. Please see Appendix C for details on how to enable the RESTful API.
Configuring the Indexer

In order to accept the data from the Forwarder, the Indexer must first be configured to receive it. This is done with a Receiver process. Enable the Receiver process at the Indexer’s command line interface with the command:

```
splunk enable listen 9997
```

Note that by convention, Splunk Forwarders use port 9997 to transfer data, but you can specify any unused port. If no authenticated session to the Indexer already exists, the Indexer will prompt for a username and password.

By setting up the Receiver, the Indexer will automatically add the data inputs for the files or streams it receives.

![Add new Receiver](image)

As shown in Figure 2, you can also define the Receiver from the Indexer’s web GUI. Under the Settings menu, in the Data section, choose “Forwarding and receiving”. Under “Receive data”, click “Add new”, then enter the port number of the Receiver.

Once the Receiver is in place, it will automatically ingest the data pushed to it by the Forwarder.
Installing the Universal Forwarder

The high-level steps for installation and configuration of the Linux forwarder are:

1. Download and install the Splunk Universal Forwarder on the Linux machine according to the Splunk documentation. For Oracle Linux, use the RPM file for your architecture and install it with `yum` or `rpm`. The Forwarder may run as any user on the system. If it is to be run as a non-root user, ensure that the user has the appropriate permissions to read the inputs that will be specified.

2. Set the environment variable `$SPLUNK_HOME` to the directory the Forwarder was installed to. If the Forwarder is installed from an RPM package using its defaults, this will be the `/opt/splunkforwarder` directory. Append the `$PATH` variable to the location of the `splunk` executable, then start the Splunk Forwarder. The export lines may be added to your shell’s initialization script to make them permanent:

   ```bash
   export SPLUNK_HOME=/opt/splunkforwarder
   export PATH=${PATH}:${SPLUNK_HOME}/bin
   splunk start
   ```

3. The default Splunk admin password should be changed with the following command. If the CLI session on the Forwarder has not yet been authenticated, the screen interface will prompt you for a username and password.

   ```bash
   splunk edit user admin -password <New_Splunk_Admin_Password> \ -role admin
   ```

4. Configure the Forwarder to point to a specific receiving Indexer. Specify the host by either IP address or name. The default port for Splunk communication is 9997, but you can easily change this at the Indexer end.

   ```bash
   splunk add forward-server <host>:<port>
   ```

The Splunk Forwarder package does not have a web interface. You must configure it from the command line or through configuration files.

---

3 As of this writing, the documentation is available at [http://docs.splunk.com/Documentation/Splunk/latest/Forwarding/Deployanixdfmanually](http://docs.splunk.com/Documentation/Splunk/latest/Forwarding/Deployanixdfmanually)
Collecting Logs from the Oracle ZFS Storage Appliance

Log files are a rich source of data for Splunk, and the Oracle ZFS Storage Appliance logs a great deal of data. This data is distributed into five different logs: alert, audit, fault, phone-home and system. Only the system log is presented through the standard syslog protocol, which has the ability to forward the system log to another server.

The remainder of these logs can be collected from the Oracle ZFS Storage Appliance through other mechanisms, including the Browser User Interface (BUI), the Command Line Interface (CLI), and the Representational State Transfer (REST) interface. The REST interface is the most flexible and is used in this white paper for all interaction with the Oracle ZFS Storage Appliance. The REST interface returns the logs in the JavaScript Object Notation (JSON) format, which is easily parsed by Splunk.

Three steps are needed to bring the logs into Splunk. The first of these is log collection, which is done with a script.

The get_appliance_logs.py Script

The get_appliance_logs.py script that is listed in Appendix A is an example of how the logs may be collected by a script that is designed to be run periodically on a Forwarder node.

The usage of get_appliance_logs.py as displayed in the script is:

```
usage: get_appliance_logs.py [parameters] <zfssa-host>
parameters:
  -u <username> : Login user on appliance.
  -p <filepath> : The local filepath the logs will be written to.
  -t <logtype>  : Type of log to collect. Valid types are: alert, audit, fault, phone-home and system
  -F : If -F is given, filter login and logout entries. Only applicable if logtype is audit
All parameters except -F are required.
```

The script, when run, prompts for a password for the Oracle ZFS Storage Appliance username passed on the command line. It is highly recommended to use a non-privileged user to query the REST interface. Such a user can be created through the Oracle ZFS Storage Appliance BUI. When creating the user, do not apply any roles to the account. (Please see Appendix B for an example of adding such a user.)

The script then authenticates itself to the Oracle ZFS Storage Appliance, gathers the entries for the specified log type that have been generated since the last time the script was run, and writes a file of the form `<zfssa-host>.<logtype>.log` to a designated path on the Forwarder, rotating out the previous log files in a manner similar to the Linux `logrotate` command.

When the audit log is requested, a special filter can be applied to prevent writing each session login or logout from the BUI, CLI or REST interfaces. These entries comprise the bulk of the audit file and may not be useful to pass to Splunk for indexing.

---

4 Please refer to the Oracle ZFS Storage Appliance documentation for details on each log type.
One of the issues with collecting the log data through a Splunk Forwarder is that the original source of the data can become conflated with logs from other Oracle ZFS Storage Appliances. In order to be able to search on logs from a single Oracle ZFS Storage Appliance, the script adds a field named “host” to each log entry to correctly identify its origin.

The script should be installed in an appropriate location such as `/usr/local/bin` and be given execute permissions with the command `chmod +x get_appliance_logs.py` The designated file path that will be used by the script to write the logs to must exist and be writable by the user running the script.

With the script in place, the second step is the configuration of the Forwarder.

Configuring the Forwarders to Parse Timestamps

The Oracle ZFS Storage Appliance logs use a timestamp format that is not immediately recognized by Splunk. To allow Splunk to find and parse the timestamp within the log’s JSON format, edit the file `$SPLUNK_HOME/etc/system/local/props.conf` to include the following stanza:

```bash
[_json]
MAX_TIMESTAMP_LOOKAHEAD = 300
TIME_FORMAT = %Y%m%dT%H:%M:%S
```

This stanza increases the number of characters Splunk will look into a JSON record to identify a timestamp to 300 and adds the definition of the Oracle ZFS Storage Appliance’s timestamp.

Note that if a Splunk infrastructure has multiple Forwarders collecting logs from Oracle ZFS Storage Appliances, this change must be made on each Forwarder. It is the Forwarders and not the Indexers that must have this timestamp stanza in place.

The third step that must be in place is to tell the Forwarder to read the logs.

Defining a Monitor

The method by which the Forwarder determines what to read and pass to an Indexer is called a Monitor. Because the script in the example writes logs to a file, the Forwarder must be told to monitor a directory for changes to the files located there. The `<filepath>` argument to the `get_appliance_logs.py` script becomes the `<source>` for the Monitor.

Any changes to the files in the monitored directory are noted by the Forwarder and immediately acted upon.

To add the Monitor from a file, edit `$SPLUNK_HOME/etc/system/local/inputs.conf` to include a stanza of the format:

```bash
[monitor://<source>]
sourcetype = _json
```

If the example file path is `/var/log/splunk/`, then the stanza would be:

```bash
[monitor://var/log/splunk]
```
sourcetype = _json
blacklist = \.(gz|bz2|z|zip)$

Note that the `sourcetype` must be specified as `_json` or Splunk will improperly parse the logs.

The script does not provide for compressing older log files, because Splunk’s default behavior for handling logs is sub-optimal. It understands log rotation to the extent that if a logfile has been renamed, it does not have to be reread, but if the file is renamed and compressed, Splunk will uncompress it and re-parse it, resulting in duplicate events.

If compression is used for the log rotation, adding a blacklist rule as shown in the previous code will prevent Splunk from processing any files ending with the given patterns.

To add the Monitor from the command line, use this:

```
$ splunk add monitor <source> -sourcetype _json
```

Note that the directory name must end with a slash(\/) when the command line is used.

To pick up the new configuration, the Forwarder must be restarted with the following command:

```
splunk restart
```
Using the Oracle ZFS Storage Appliance Log Data

When the script begins generating data into the monitored directory, the Forwarder will automatically push it to the Indexer. This can be seen in the “What to Search” section of the Search screen.

Clicking the “Data Summary” button brings up a list of hosts from which Splunk collects its data. Note that by default, Splunk uses as the “Host” the hostname of the Forwarder, not the hostname of the Oracle ZFS Storage Appliance.

Clicking on the Forwarder’s host name will pull in all of the records that the Indexer has collected from the Forwarder. Each of the Forwarders in the Splunk infrastructure will be listed under the Host column.

In the instances where multiple Oracle ZFS Storage Appliances are reporting to a single Forwarder, a search can be created based on all records from that Forwarder. In this example, data from a single machine reporting to the Forwarder will be reported upon.
Figure 5. Search results

Note in Figure 5 that under the “i” column there are arrows to expand the record and act upon them. Click on the “i” for the first record.

Figure 6. Expanded record
Find the field “extracted_host” and click the box to the left. The field is then moved up to the “Selected” area, making it much easier for the Indexer to sort on a single Oracle ZFS Storage Appliance. This selection is applied to all records pushed from the Forwarder. This action is persistent so it will not have to be repeated for other searches even if they are for different Oracle ZFS Storage Appliances.

Clicking the arrow in the “I” column once again will close the expanded window. Note the “extracted_host” field is now displayed as “host” in the log records. By clicking on the value of the extracted host, a list of options is displayed.

Figure 7. Showing the correct host

Example: Charting Logins

Clicking on “New search” will create a search for only that Oracle ZFS Storage Appliance. Other fields can be added to the search from the same popup menu. By clicking on the value of “User logged in,” highlighted in the text under “summary,” the search is restricted to login events.

Figure 8. Creating a search

As the search is created, the command line for it is shown at the top. It may be manually edited, such as the example that follows which adds a wildcard to the summary value to collect all logins to an Oracle ZFS Storage Appliance.
Figure 9. Editing the search

To create a graph from this information, a transforming command\(^5\) can be added to the search.

In the following example, adding “timechart count” to the search command will create a graph of the number of logins over time. Use the pull-down menus to change the style and format of the graph.

Figure 10. Visualizing the search

---

\(^5\) Transforming commands “transform” the specified cell values for each event into numerical values that Splunk can use for statistical purposes.

Transforming commands include chart, timechart, stats, top, rare, contingency, and highlight. Transforming commands are required to transform search result data into the data structures required for visualizations such as column, bar, line, area, and pie charts.
Conclusion

The provided Python script in this paper has been written to illustrate the use of the RESTful API in extracting log data from an Oracle ZFS Storage Appliance and may not provide adequate error checking on input parameters or error reporting on failed commands. Please use the examples accordingly.

When creating programs in production environments, pay proper attention to writing code that fully checks user input and provides adequate detail in diagnostic error messages for the user to fully understand the nature of a failure. A more automated method of authenticating the user to the Oracle ZFS Storage Appliance is left as an exercise for the reader.

By leveraging the RESTful API on the Oracle ZFS Storage Appliance, all logs and metrics can be imported into Splunk, allowing for a comprehensive overview of an organization’s infrastructure with the ability to drill down into details as needed.
References

See the following resources for additional information relating to the products covered in this document.

- Oracle RESTful API documentation
  https://docs.oracle.com/cd/E51475_01/html/E52433/index.html

- Oracle ZFS Storage Appliance White Papers and Subject-Specific Resources
  Including: “Working with the RESTful API for the Oracle ZFS Storage Appliance” at

- Oracle ZFS Storage Appliance Product Information
  https://www.oracle.com/storage/nas/index.html

- Oracle ZFS Storage Appliance Documentation Library, including Installation, Analytics, Customer Service, and Administration guides:
  The Oracle ZFS Storage Appliance Administration Guide is also available through the Oracle ZFS Storage Appliance help context.
  The Help function in Oracle ZFS Storage Appliance can be accessed through the browser user interface.

- Splunk information and documentation
  http://splunk.com
Appendix A: Python code for get_appliance_logs.py

#!/usr/bin/python

# Copyright (c) 2015, Oracle and/or its affiliates. All rights reserved.
#
"""Collect logs from an Oracle ZFS Storage Appliance. The logs are returned in
JSON format and will be rotated as needed."""

import pycurl
import getopt
import getpass
import json
import sys
import os
import glob
import cStringIO

def usage():
    print "Collect logs from an Oracle ZFS Storage appliance and write them "
    print "to a JSON formatted file."
    print "usage: get_appliance_logs.py [parameters] <zfssa-host>"
    print "parameters:"
    print "  -u <username> : Login user on appliance."
    print "  -p <filepath> : The local filepath the logs will be written to."
    print "  -t <logtype>  : Type of log to collect. Valid types are:"
    print "                 alert, audit, fault, phone-home and system"
    print "  -F : If -F is given, filter login and logout entries. Only"  
    print "         applicable if logtype is audit"
    print "All parameters except -F are required."

def rotate_logs(filename, keep_logs):
    """Rotate the log files in a similar manner to the logrotate command"""
    loglist = sorted(glob.glob(filename + '.*'))
    if keep_logs == 0 or keep_logs >= 1000:
        log_num_len = 3
        keep_logs = 1000
    else:
        log_num_len = len(str(keep_logs - 1))
    latest = len(loglist)
    for myfile in reversed(loglist):
        logstr = '.' + str(latest).zfill(log_num_len)
        os.rename(myfile, filename + logstr)
        latest -= 1
    os.rename(filename, filename + '.' + str(latest).zfill(log_num_len))
    return

def find_last_logs(filename):
    """Find the last time a log entry was written to the extant logfile and
return all entries with that timestamp."""
    try:
        with open(filename, 'r') as fp:
            oldlog = json.load(fp)
        fp.close()
        lastlogs = []
        last_time = oldlog[-1]['timestamp']
        for entry in reversed(oldlog):
            if entry['timestamp'] == last_time:
lastlogs.append(entry)
    return lastlogs
except IOError:
    return None
except AttributeError as err:
    print "Error ", err, " opening ", filename
    sys.exit(6)

def get_logs(client, urlbase, logtype, last_time):
    buf = cStringIO.StringIO()
    # Do not verify appliance self-signed certificate
    client.setopt(client.SSL_VERIFYPEER, False)
    client.setopt(client.SSL_VERIFYHOST, False)
    client.setopt(client.FAILONERROR, True)
    client.setopt(client.WRITEFUNCTION, buf.write)
    if not last_time:
        urlfull = '{0}/api/log/v1/logs/{1}'.format(urlbase, logtype)
    else:
        urlfull = '{0}/api/log/v1/logs/{1}?start={2}'.format(urlbase,
                                                             logtype,
                                                             last_time)
    client.setopt(client.URL, urlfull)
    try:
        client.perform()
    except pycurl.error, error:
        errno, errstr = error
        print 'An error occurred: ', errstr
        print 'URL: ', client.getinfo(client.EFFECTIVE_URL)
        client.close()
        sys.exit(3)
    else:
        # The appliance returns the logs in a dictionary with a single key:value
        # pair, with the key of 'logs' and a value of a list of all the
        # pertinent entries. We only need the list.
        client.close()
        return json.loads(buf.getvalue())['logs']

def main(argv):
    # User-defined variables - These may be changed for your environment.
    # Define how many logs are to be kept, up to 1000. Zero is the same as 1000
    keep_logs = 15
    # End of user defined variables

    password = getpass.getpass()
    user = ''
    logpath = ''
    logtype = ''
    do_filter = False
    looped_logs = False
    try:
        opts, args = getopt.getopt(argv[1:], "p:t:u:F")
    except getopt.GetoptError as err:
        print str(err)
        usage()
        sys.exit(1)
    for opt, arg in opts:
        if opt == "-p":
            logpath = arg
        elif opt == "-t":
            logtype = arg
elif opt == "-u":
    user = arg
elif opt == "-F" and logtype == "audit":
    do_filter = True

if len(args) != 1:
    print "Insufficient arguments "
    usage()
    sys.exit(2)

host = args[0]

urlbase = "https://{0}:215".format(host)
filename = '{0}/{1}.{2}.log'.format(logpath, host, logtype)

# Find the last log entry written, if any
if os.path.exists(filename):
    last_logs = find_last_logs(filename)
    if len(last_logs) != 0:
        last_time = last_logs[0]['timestamp']
    else:
        last_time = None
else:
    last_logs = []
    last_time = None

# Fetch the logs from the host
client = pycurl.Curl()
client.setopt(client.USERPWD, "{0}:{1}".format(user, password))
logs = get_logs(client, urlbase, logtype, last_time)
if not logs:
    print "No " + logtype + " logs found"
    sys.exit(4)

# Compare the last log entries written with the new log entries
# collected and discard any matches in the new log
for test_entry in last_logs:
    looped_logs = True
    for log_entry in logs[:]:
        log_entry["host"] = host
        if cmp(test_entry, log_entry) == 0:
            logs.remove(log_entry)
            continue

if do_filter:
    for entry in logs[:]:
        try:
            if entry['summary'].find("User logged") != -1 or \
                entry['summary'].find("Browser session") != -1:
                logs.remove(entry)
        except KeyError:
            print "key error on entry " + str(entry)
            sys.exit(5)

# Add the appliance field to the log record if it wasn't done in checking
# the last log entries
if not looped_logs:
    for entry in logs[:]:
        entry["host"] = host

if len(logs) > 0:
    # Only rotate logs and write a new log file if there are new entries
    if os.path.exists(filename):
        rotate_logs(filename, keep_logs)
    try:
        with open(filename, 'w') as fp:
            json.dump(logs, fp, sort_keys=True, indent=2)
fp.close()
except IOError as e:
    print 'I/O error({0}) writing {1}: {2}'.format(e.errno, filename,
                 e.strerror)

if __name__ == "__main__":
    main(sys.argv)
Appendix B: Adding a Non-Privileged User to the Oracle ZFS Storage Appliance

The Oracle ZFS Storage Appliance does not require the user logging into the RESTful API to have any administrative privileges if it is only reading data. A new user without these privileges should be created to access the RESTful API.

To create this user, log in to the Oracle ZFS Storage Appliance’s web browser user interface (BUI). Log in as a privileged user such as root. Click on Configuration, then on Users. Click on the plus sign above the list of users to get to the Add User screen.

Start by clicking the button to make this user “Local Only”. Enter the name of the new user. This example uses the username “reporter”. Enter a full name for the user if desired, then enter the password twice.

Finally, ensure that none of the roles at the bottom of the window are checked off. Assigning no roles to the user ensures that no privileges will be assigned.
Appendix C: Enabling the RESTful Interface

Before the RESTful interface on the Oracle ZFS Storage Appliance can be accessed, it must be enabled through the browser user interface (BUI). Log in to the BUI and click on the Configuration menu item. The Services page will be displayed. At the bottom of the page is a line in the Remote Access section labeled REST. If the indicator is not green, click on the on/off button as shown in Figure 13. This button toggles the REST interface.

Figure 13. Enabling the REST interface