Centralized Logging in a Decentralized World

JAMES DONN AND TIM HARTMANN

James Donn has been working as a Senior Network Management Systems Engineer (NMSE) for the past four years and is responsible for managing and monitoring both network devices and servers. Before this, he worked as an NMSE with HSBC Bank, USA.
james_donn@harvard.edu

Tim Hartmann's work at Harvard University includes support for networking, systems, and services operations. Tim's current focus is on automated application and systems deployment using DevOps as guiding principles. He is also involved in virtualization and directory services.
tim_hartmann@harvard.edu

In 2008, we were both working for the Faculty of Arts and Sciences at Harvard University when it became clear that we needed a better way to manage our syslog environments. We both needed a centralized repository for logs that was easily searchable, scalable, and had the ability to produce graphs and reports. We began experimenting with Splunk, and over time we added more and more log sources. Today, we have a vastly improved log management and reporting system, which was the result of a few phased learning periods, with a couple of surprises along the way.

While we quickly realized the benefits of combining our Network and Systems syslog architectures, sharing a common architecture in this fashion was a first for our groups. We were presented with a few challenges right away:

1. Approval to act on this from our management teams, which historically did not collaborate in this fashion
2. Role-based log separation
3. Tandem administration of a single application

We were given the green light to proceed, and a new era of collaboration began. We chose Splunk shortly after running a quick proof of concept within our environment because we found it to be simple, scalable, and extremely useful.

Splunk runs as a server-side application with two major components. The most resource-intensive portion of the application is indexing, which involves converting raw event data into searchable events and storing them. The other major component is searching. Searches can be scheduled to run at specified intervals and take action when criteria, such as event count, have been reached. The actions that you can take include sending emails, updating RSS feeds, and executing scripts. Searches can be very lightweight, involving a small amount of data, or may take several minutes to complete while searching a month's worth of data. Therefore, your server load from scheduled searches can vary greatly depending on your implementation and the frequency of searches.

A few years into the Splunk implementation and after attending Splunk events and speaking with other Splunkers, we initiated a common phased approach to implementing the application:

Phase 1—"Just get them in!"
Phase 2—More logs and field extractions
Phase 3—Indexes and agents
Phase 4—Building custom applications

We will look more closely at each of these phases and share some of our experiences during each phase.

Phase 1—"Just get them in!"

Our first step when deploying our centralized logging repository was to gather as much data as possible and test the application's ability to process information. Since both Network and Systems teams already had their own independent installations of syslog-NG, this was fairly painless. We simply relayed our logs from the syslog-NG servers to the new Splunk server, using unique relay ports to separate data.

This allowed us to leverage the source of the logs to configure group-based user roles and permissions and to share a diverse log pool with Network Engineering, Systems Administration, Research Computing, Application Development, and Network Operations teams.

This quick "just get them in" strategy worked very well with Splunk, due to the way that the application indexes logs and extracts fields. When logs are first received in Splunk, they are indexed, and only a small number of fields, such as hostname and time, are extracted. All other fields are usually extracted at search time. This allows you to build custom field extractions after you have collected logs, without having to wait for new logs to come in to see the changes. Other applications follow a different philosophy, where all incoming logs must fit predefined templates or require custom templates before you can use the data. Both approaches have their advantages and disadvantages.

The disadvantages to post-processing logs:

- There are no canned reports to start with.
- Becoming familiar with the application and Splunk's native commands involves a learning curve.
- You will be reinvesting time into the application as your customer base grows.

The advantages to post-processing the logs:

- You can get started very quickly.
- It allows for greater flexibility when determining exactly what you want to do with the logs.
- You can change field extractions at any time, without having to worry about your repository.
- You can build very customized reports and alerts.

After getting the logs in, we set up a dozen rules to look for critical logs and forward them as SNMP traps to our event management systems. This ultimately replaced our legacy syslog adapter into the alerting infrastructure, which was much more difficult to configure and maintain.

Use the Splunk API!

One lesson that we learned in this phase is that the storage location of the raw data changed during software upgrades. Since our searches could return results for many devices, we needed to evaluate the results before sending a trap for each, possibly unique, host. These changes forced us to update our notification scripts a few times before we started to use Splunk's RESTful API, which has yet to change.

High-Level Architecture

Initially, our infrastructure consisted of a mirrored set of servers running Red Hat Enterprise Linux on commodity hardware, with an internal YUM repository to maintain software deployment and updates. Keeping our architecture simple allowed us to expand easily and spend more time getting useful information out of the application, rather than wrestling with the nuances of application management.

Licensing

The application license is based on the amount of data that is processed (indexed) each day. To ensure that we purchased enough to allow for growth, we combined the sizes of our daily syslog files and multiplied by 2. This 20 GB/day license would give us enough room to double the size of our initial footprint.

Our Views Changed

Now that we had collected all of our data, we wanted to see if we could get a panorama of events that we might have missed. In some ways our methodology for viewing logs started to shift. In our old-world view, we would run large log files (5+ GB) through a grep pipeline to pull out interesting events and build reports. Often, we would have to request that other teams perform similar searches against their logs, and manually correlate the results. In contrast, using our new workflow we found ourselves viewing larger amounts of combined data, obtaining summary information, and correlating events much more quickly.

Our initial deployment ran very well for over a year before we started to look into other ways that we could significantly improve our use of our logging engine. We already had what we set out to create: a simple and useful, low-maintenance tool.

Phase 2—More Logs and Field Extractions

More Logs!

At this stage, we had an easy-to-use search interface to all of our logs with some automated alerting. It was immediately evident to all of the engineers that worked on multiple devices concurrently that "more was better." We started getting requests to assist in getting more devices into the application, as well as to expand to different types of logs.

One challenge we encountered was adding logs from a proprietary application, which did not log to the system's syslogger. This forced us to look at using Splunk as an agent. Neither one of us liked the idea of agents in general; large-scale agent management and concerns about potential resource constraints drove our opinions. However, we installed a few agents to scrape log files for our edge cases where we could not send them via syslog.

Fortunately, Splunk made agent installation and configuration relatively easy. We manually managed the agents, and they proved to be very well behaved. One of the side effects that we saw from the installation of our initial agents was a dramatic increase in our volume of log data. We were processing approximately 10 GB of syslogs per day from 3,500 network devices and 400 servers. With the addition of three agents, we added an additional 8 GB of data indexed per day. While the data gained from forwarders was excellent, it is an important consideration when attempting to estimate growth and planning for future license requirements. Since these results were unexpected, we had to increase our license from 20 GB to 50 GB.

Now that our new centralized logging model had become well rooted, we collapsed our syslog-NG servers together. At the same time, we also started to forward logs using TCP rather than UDP. When making this change in syslog-NG, we found that we were no longer able to spoof the sending address. Therefore we had to update Splunk to look into the logs to obtain the proper hostname.

Field Extractions

Next, we set up additional field extractions. After running Splunk hands off for over a year, we started investing time into reconfiguring it. We wanted to enhance our ability to troubleshoot issues, create more meaningful alerts, and build better reports. Custom field extractions allowed us to define portions of the logs, using regular expressions, which became pivot points for searches and reporting. For instance, let's look at a standard Cisco syslog:

```
2011.01.20 11:51:32 switch123.harvard.edu local7 notice 65: Jan
20 11:51:31.540: %LINEPROTO-5-UPDOWN: Line protocol on Interface
GigabitEthernet1/0/14, changed state to up
```

To make sense out of the 500,000+ messages like this that we receive every day, we created the following field extractions:

```
Host:          switch123.harvard.edu
Facility:      local7
Priority:      notice
Message_type:  %LINEPROTO-5-UPDOWN:
Message: Line protocol on Interface GigabitEthernet1/0/14, changed state to
up
```

We can now determine which is the chattiest message type over the past 24 hours with a single click after the search. Visualizing trends in log volume for a specific event has proven to be very useful. For instance, seeing a large spike in failed SSH attempts against a single host is something you might want to set up an alert for.

Phase 3—Indexes and Agents

Indexes

As our Splunk infrastructure expanded to 50 GB per day, we outgrew our single-index solution. We were simply putting too many logs into a single index, which made our searches take longer. While you could search different indexes with any revision of the software, the indexes needed to be explicitly declared at search time. We thought that this was too much overhead for our users. However, the next version released allowed a user to automatically search any index that they had permissions for.

With this new update, we migrated our source-based separation of logs to an index-based separation. Since indexes store data in different folders on disk, we gained security with data separation, as well as speeding up our searches by reducing the number of logs searched.

Agents

After proving that Splunk agents were stable and reliable, we placed one on our syslog-NG servers to scrape all of the syslogs that they collected. Instead of forwarding the logs via TCP, we directed the logs to a file, which gets zeroed out once a day. With a simple configuration to the Splunk agent, we were then sending logs to various indexes. This change also came with a few extra benefits.

If there are any communications issues between the Splunk agent and the indexing servers, the data is held until communications resume. This means that we could now restart our Splunk indexing servers without having to worry about the loss of data while the application was not available. These same logs would have fallen into the bit bucket using any other forwarding strategy.

Agents can also be configured to deliver data that they obtain to the indexers over SSL. As we tuned our implementation, we wanted to encrypt all data as close to the source as possible. Since there are devices that you will never be able to accomplish this on (e.g., network devices), the agent on the syslog-NG server gave us the best results.

Splunk Applications and Agents

Free Splunk applications also came with this software upgrade. These applications are really a series of predefined Splunk searches, charts, and dashboards. When testing them out, we found the UNIX and Windows applications to be extremely useful. They allow you to see details of disks, network interfaces, process tables, etc., at a glance. However, to obtain this information, the agent is required to collect it via "scripted inputs." Scripted inputs log the data that is returned after running a specified command or script, such as "df –h", "netstat", "ps –ef", etc.

As you can imagine, having the historical output from these commands, running every $x seconds, has been an enormous benefit in troubleshooting performance-related issues. Every aspect of a Splunk agent is configurable, including the intervals at which the data is collected and which commands are run. We have a few agents collecting data from Expect scripts run against network devices, which allow us to gather data that SNMP cannot provide.

After using the UNIX and Windows apps to resolve a few problems, we realized that we needed agents on every box to gain this level of visibility on our servers. Although Splunk has a deployment application to manage Splunk agents, it is really a Splunk agent configuration tool. It does not upgrade versions of the Splunk agents or ensure that they are installed on any new servers. To address this issue, we used a combination of Puppet and subversion. Puppet ensures that the agent is installed, running, and configured. In addition, Puppet checks out custom Splunk applications directly from version control in order to manage Splunk indexers and search heads. This combination of tools not only mitigates risk but also allows us to approach Splunk application development in a modular fashion.

Phase 4—Building Custom Applications

Now that a few team members had deep knowledge of the application and search language, they had become the point persons for all issues that required "power searching." This represented most of the problems that were incurred on a daily basis. To ensure that anyone could perform more simplified guided searches, we began building small applications. Not only did this free up more resources, but more people became comfortable using the application to solve problems.

Some of the basic applications include:

- Form search for email transactions
- Dashboards to show which machines currently live on a particular Xen server
- Saturation graphs for particular VLANs

After attending a Splunk developers training, we committed to building many more applications in-house. We have a few applications in development that we are very excited about. We are currently developing a "manager of managers" application to serve as the integration point for all data from various monitoring tools, to perform custom correlations, and to act as an event notification system for all alerts.

Architectural Options

Our original architecture was very simple and met our needs for the 20 GB of data that we planned to process per day. However, organic growth and organizational changes increased our potential daily log indexing towards 200 GB per day. With a log volume of this size, we needed to reevaluate our architectural options.

The new architecture design (see Figure 1) includes four stand-alone indexers and separate search heads per side of a mirrored implementation. The stand-alone indexer pool will receive data, which is automatically distributed across all of the indexers. Splunk then uses MapReduce technology to expedite the searches across the indexes. Use of various search heads allows us to expand our customer base to include groups that use different types of authentication, to reduce the overhead on a single server for scheduling searches, and to develop applications for any Splunk administrative group. For example, our Security team can now provision their own users and develop their own applications, which searches a common dataset in the indexing pool. This modular architecture has allowed us to collaborate with several groups at the University.

This model can be leveraged for horizontal growth, which allows us to scale exponentially. When we see that we are going to need to expand the amount of data that we index, we simply add more indexing servers to the pool of indexers. We can also add more search heads as our customer base grows and we find that scheduled searches are starting to overwhelm the existing servers.

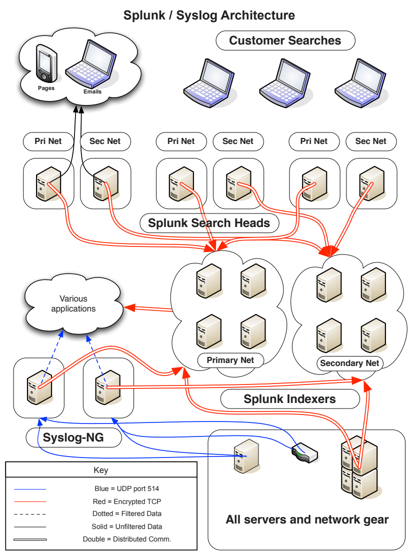

Figure 1: Splunk/Syslog architecture

Figure 1 is a diagram of the architecture that we are currently building out. Our strategies from the bottom up are to:

- Send all data encrypted from the source where possible
- Leverage distributed indexing cloud, using MapReduce technology
- Isolate customer environments from each other to:
    - - Increase security (data and user separation / isolation)
    - - Allow for multiple types of authentication interfaces
    - - Allow independent application development
    - - Increase search performance
- Duplicate vertical environments to ensure the highest degree of availability

Conclusion

We first set out to solve some very simple problems with Splunk. As we learned more about the application and it evolved, we have become much more dependent on it. Splunk has since taken a unique place in our environment. It has become a stable and flexible platform that provides solutions for an extremely broad range of problems. In many places, it has become the glue that fills in the gaps missed by other monitoring applications.

Not only is it a great tool to obtain, parse, and search data with, it is fantastic at providing data visualization, custom applications, and integration with other tools. While it has been referred to as "grep for your logs," it also acts like "awk for your logs," allowing you to pivot reports and searches on any custom field. Splunk also provides a powerful search language to perform analysis with commands, which are very similar to common UNIX tools. Splunk also follows the UNIX model of pipelines, allowing you to pass search or command results into other series of searches, internal commands, external scripts, or other tools.

The open and modular design has allowed us to use Splunk as an engine to power custom tools such as a Puppet dashboard, a MAC address tracker, security tools for forensics and compliance, and a trending tool for network, systems, and applications. While it is an extremely simple tool to start using on day one, it is also an application with great depth. We feel that these are the things that really make Splunk uniquely useful and that drive its continuing growth within our organization.