LLNL-TR-670919

Lawrence Livermore National Laboratory

Continuous Security and Configuration Monitoring of HPC Clusters

Hector D Garcia-Lomeli

Science Undergraduate Laboratory Internship (SULI)

Sacramento State University

Lawrence Livermore National Laboratory

Livermore, CA

May 8, 2015

Prepared in fullfillment of the requirements of the Department of Energy's Science Undergraduate Laboratory Internship under the direction of David Fox and Adam Bertsch in the Livermore Computing Division at Lawrence Livermore National Laboratory.

Disclaimer

Lawrence Livermore National Laboratory is operated by Lawrence Livermore National Security, LLC, for the U.S. Department of Energy, National Nuclear Security Administration under Contract DE-AC52-07NA27344.

Table of Contents

Abstract

Continuous Security and Configuration Monitoring of HPC Clusters. HECTOR D GARCIA-LOMELI (California State University, Sacramento, CA 95819), DAVID FOX, AND ADAM BERTSCH (Lawrence Livermore National Laboratory, Livermore CA, 94550)

Continuous security and configuration monitoring of information systems has been a time consuming and laborious task for system administrators at the High Performance Computing (HPC) center. Prior to this project, system administrators had to manually check the settings of thousands of nodes, which required a significant number of hours rendering the old process ineffective and inefficient. This paper explains the application of Splunk Enterprise, a software agent, and a reporting tool in the development of a user application interface to track and report on critical system updates and security compliance status of HPC Clusters. In conjunction with other configuration management systems, the reporting tool is to provide continuous situational awareness to system administrators of the compliance state of information systems. Our approach consisted of the development, testing, and deployment of an agent to collect any arbitrary information across a massively distributed computing center, and organize that information into a human-readable format. Using Splunk Enterprise, this raw data was then gathered into a central repository and indexed for search, analysis, and correlation. Following acquisition and accumulation, the reporting tool generated and presented actionable information by filtering the data according to command line parameters passed at run time. Preliminary data showed results for over six thousand nodes. Further research and expansion of this tool could lead to the development of a series of agents to gather and report critical system parameters. However, in order to make use of the flexibility and resourcefulness of the reporting tool the agent must conform to specifications set forth in this paper. This project has simplified the way system administrators gather, analyze, and report on the configuration and security state of HPC clusters, maintaining ongoing situational awareness. Rather than querying each cluster independently, compliance checking can be managed from one central location.

Keywords: *Software Agent, Node, Splunk Enterprise, Syslog*

Introduction

On November 24, 2014 Sony Pictures Entertainment (SPE) suffered a major security breach of its information systems. Unauthorized users gained access to its internal network destroying and stealing roughly 100 terabytes of confidential and intellectual property, which was then leaked to the public at various Internet sites. This incident prompted the White House to call it "a serious national security matter."[1] The Sony hack and similar incidents that occurred in 2014 served as a wakeup call, shedding light on the ever increasing number and complexities of Advanced Persistent Threats (APT) and the need for an organization to protect and secure its critical information assets and computer systems infrastructure. Similarly, in September 2014 The Home Depot reported a breach of its information systems used at its retail stores. Sony, The Home Depot, Target and Bank of America are only a handful of companies from the entertainment, retail, and banking industries that have fallen victim of cyber criminals.[2]

The current underlying challenge for government and private companies alike are to develop an adequate, comprehensive, reliable, and cost-effective information systems security framework. The security scheme must adhere to guidelines set forth by the National Institute for Standards and Technology (NIST) published under Special Publication (SP) 800-137, "Information Security Continuous Monitoring for Federal Information Systems and Organizations." Such scheme must comply with the Federal Information Security Management Act (FISMA) mandate and Department of Energy Order 205.1B, "Department of Energy Cyber Security Program". Given the dynamic nature of cyber space, attackers have become more sophisticated in their approaches and tactics rendering current computer security systems, procedures, and policies ineffective. "End points, network devices, systems and applications, and security tools and processes that passed a FISMA compliance checklist yesterday could become noncompliant in the blink of an eye." [3]

A comprehensive computer security strategy involves specification (policy), implementation (mechanisms), and correctness (assurance). [8] "Failure to patch a system, install a security update, or prohibit a user from downloading an application can all suddenly render secure systems vulnerable…[as a result] a more continuous view into the state of devices and applications to maintain a more continuous compliant state" is paramount. [3] The end goal is the implementation of an effective and reliable continuous monitoring framework that would give system administrators, security managers, and auditors near real-time information of the security state of data, the network, endpoints, and applications.[3] Although there are many papers focusing on the subject of continuous monitoring, this paper focuses on system information and log collection in the implementation of the *end client agent*, *Splunk Enterprise*, and *Splunk Software Development Kit (SDK)* for *Python* to design, develop, test, and deploy a continuous security and configuration monitoring framework on the High Performance Computing clusters to support systems used in Livermore Computing (LC) Center.

Software Tools and Methods

This project was broken down into three major design tasks which included developing and deploying a *"logger"* agent across system resources, utilizing Splunk to capture, index, and correlate logged messages in a searchable repository, and developing an application user interface (API) in Python to interact with the Splunk search engine to collect, analyze, and display results.

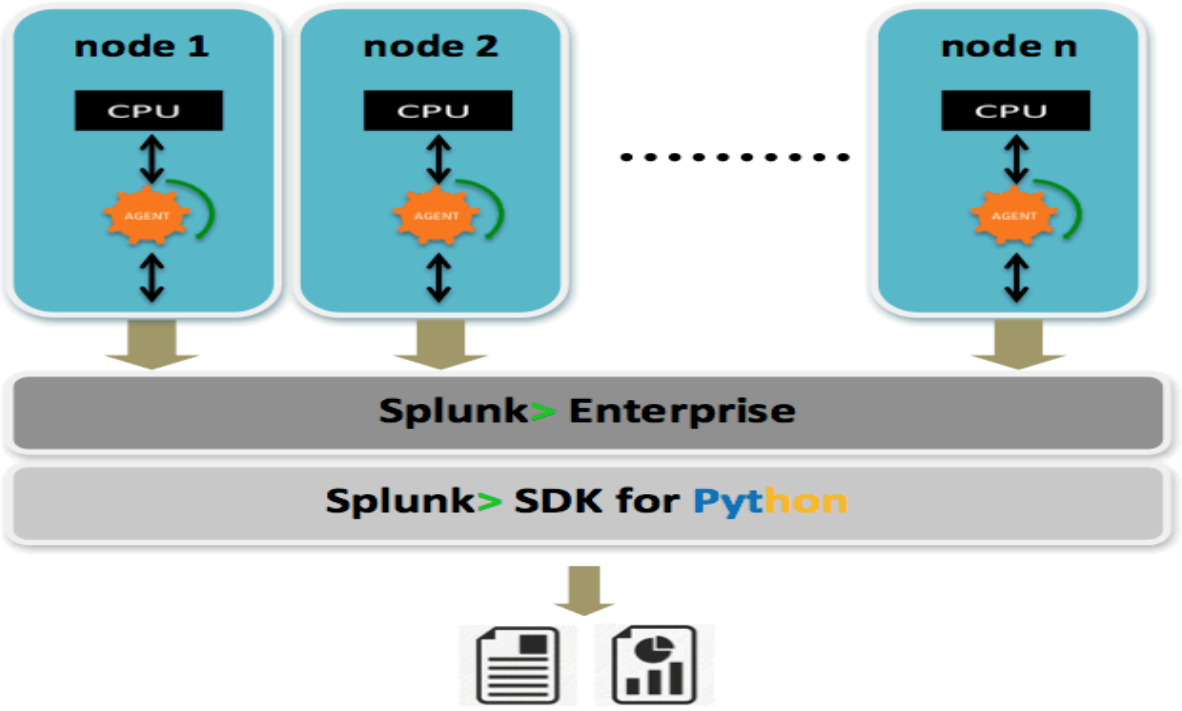

Figure 1 – Continuous monitoring high-level operational concept, which consists of the end client agent running in every node, Splunk Enterprise, and Splunk DSK for Python

Implementation of the "Agent"

Implementing and deploying a bash shell script to record security and configuration information from various sources accomplished continuous log collection. The agent provided current information reflecting the security and configuration state of HPC clusters. In this context, the "agent" refers to a simple implementation of the *logger* shell utility to write a formatted message to specific system log files. The agent was deployed across over thirty HPC clusters and over six thousand nodes. It was configured to gather and record information about the test being executed, test status to include OS version and type as well as kernel version and

type. Table 1 shows a list of common field-value pairs as reported by the agent. These fields reveal important information about the system's security and configuration states; therefore, giving system administrators and auditors near real-time information about the true state of the system. Although the application of the logger utility could be extended to notify or warn system administrators of certain events, its scope in this project was limited to monitoring the security and configuration states of its host. The events and states logged by the agent served as inputs to the Splunk engine.

Field	Subfields	Description	Example
test_id	*None*	*Unique test identifier*	• *osvers_004*
state	*None*	*State of the test: fail, pass, or other general information*	• *info/pass/fail*
info	○ *osvers* ○ *kernel* ○ *ktype* ○ *ostype*	▪ *OS version* ▪ *Kernel version* ▪ *Kernel type* ▪ *OS type*	• toss-release-2.3-3.ch5.3 • *2.6.32-431.29.2.1chaos.ch5.2.x86_64* • *Linux* • *GNU/Linux*

Table 1 – Information captured by the agent and recorded to syslog in a predefined format

Using the Splunk platform to identify, collect, analyze data, and display results

Splunk was used to gather the event data generated by the agent from all reporting nodes. At this point, a general overview of how Splunk operates is given. First, Splunk brings data in via its *forwarders* from diverse locations, divides it into events, and identifies some default fields. Second, it indexes data by creating a time-based map of the words in the data without modifying the data itself. Third, it divides a stream of machine data into individual events and centralizes them into indexes. By default, each event is composed of four fields, which include *source, sourcetype, host,* and *time* in addition to the fields reported by the agent. Finally, a *search head* distributes the *search* across many indexes consolidating and displaying the results. Splunk uses a series of command and arguments to retrieve relevant data from its indexes. A typical Splunk query may look like the following string: ***search-args | command1 command-args | command2 command-args | ….***.[10] Search commands and arguments allowed us to extract, filter, and analyze the results. For this, we built a generic Splunk query to allow our reporting tool to programmatically interact with the Splunk engine and gather only relevant data as reported by our agent.

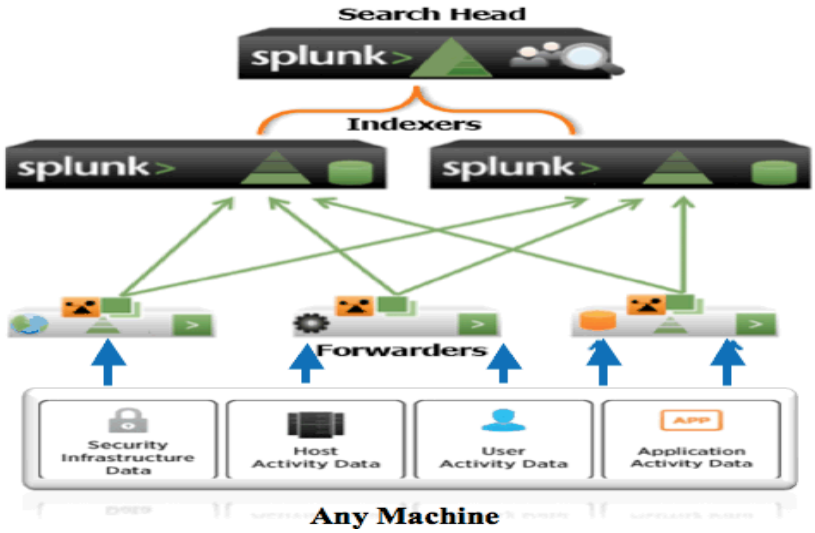

Figure 2 – High-level overviews of the operation of Splunk components to collect, centralize, index, and make data searchable [11]

The Splunk Software Development Kit (SDK) for Python served as a basis for the development of our reporting tool to conduct searches, manage Splunk configurations and logging, integrate search results into our applications, and present customized results.

Development of the "Reporting Tool" using Splunk SDK for Python

To programmatically interact with the Splunk server, we developed a series of scripts utilizing the Splunk SDK for Python as the basis for our reporting tool. The reporting tool is the interface for logging and querying Splunk, filtering and displaying the results. It takes a series of predefined command-line options and arguments, parses them using the *"argparse"* [5] module, filters, and displays the results. The results are analyzed and answers displayed in a series of reports. Table 2 provides a summary of all the reports that can be obtained using the tool.

Short/Long Flag	Argument	Description
NA/ --test_id	TEST_ID	Summarizes test_id's by state and missing/extra nodes
NA/ --host	HOST	Filters results by "host"
NA/ --ifield	arg_one/"arg two"	Displays information about arg_one. Space-delimited arguments must be enclosed with quotes, "arg two".
NA/ --info_list	NONE	Display list of info subfields
NA/ --time_range	TIME_RANGE	Returns matching events within this time range
-i/ --info	NONE	Display info subfields and their values
-l/ --list-test_id	NONE	Displays a list of all tests executed
-n/ --hostlist	NONE	Consolidates host list by range (hype1,hype2 = hype[1-2])
-v/ --verbose	NONE	Increase output verbosity

Table 2- Summary of flags and arguments available to the reporting tool

First, a secure connection was established with the *splunk* server. Login credentials for connecting to the server were hard-coded and saved into a text file called "*.splunkrc*" in the user's home directory. This file contains host, port, and version information of the Splunk instance as well as username, password, and access scheme as seen in figure three.[5]

```
# Splunk host (default: localhost)
host=localhost
# Splunk admin port (default: 8089)
port=8089
# Splunk username
username=admin
# Splunk password
password=changeme
# Access scheme (default: https)
scheme=https
# Your version of Splunk (default: 5.0)
version=5.0
```

Figure 3 – Format of the "*.splunkrc*" file[5]

Second, a generic query was built using Splunk's search commands and arguments. The following search string was used to retrieve field-value pairs as reported by the agent: ***"search test_id=testID | fields raw, host, test_id" –time_ramge=-1d***. In this case, *testID* is replaced by the unique test identifier, which could vary depending on the information that is being collected by the agent. On the other hand, the *fields* command tells Splunk to only keep the results obtained for the *raw, host,* and *test_id* fields. For more information about Splunk's internal fields, refer to Splunk Enterprise Knowledge Manager Manual.[9] The default time modifier, *--earliesttime=-1d,* sets the time range for a search. In our case, Splunk is collecting events that occurred within the last day or 24 hours. Third, the results obtained from the generic query were filtered and saved into a two-dimensional list. This list was fed as a parameter to other functions further refining and presenting the results according to the command line arguments entered by the user. Finally, the results were displayed using Python's *"tabulate"* [6] module.

Results

Capturing, consolidating, and indexing logs from over six thousand nodes, in real-time, allowed for the analysis and reporting of data providing insightful information about the state of critical OS and security updates. The agent developed and deployed for this project is called *osvers_004* and was configured to collect, track, and report important OS updates. Table 3 shows an excerpt from a list of results obtained when this test was executed.

Results 100.0% | 12241 scanned | 6084 matched | 6084 results
Verbose report for test_id = osvers_004

host	test_id	state	indextime	time	info
zwicky-lcy-oss15	osvers_004	info	1430229063	2015-04-28T06:51:01.000-07:00	osvers=toss-release-2.3-3.ch5.3,kernel=2.6.32-504.12.2.2chaos.ch5.3.x86_64,ktype=Linux,ostype=GNU/Linux
vulcanlac3	osvers_004	info	1430218027	2015-04-28T03:47:06.000-07:00	osvers=Red Hat Enterprise Linux Server release 6.5 (Santiago),kernel=2.6.32-431.23.3.1chaos.2blueos_small_pages.bl12.2.ppc64,ktype=Linux,ostype=GNU/Linux
zmist1	osvers_004	info	1430227906	2015-04-28T06:31:46.000-07:00	osvers=UNKNOWN,kernel=3.16.0-30-generic,ktype=Linux,ostype=GNU/Linux
hype223	osvers_004	info	1430220664	2015-04-28T04:31:02.000-07:00	osvers=toss-release-2.3-2rc1.ch5.3,kernel=2.6.32-504.12.2.2chaos.ch5.3.x86_64,ktype=Linux,ostype=GNU/Linux

Table 3- Summary of host's statistics by OS and kernel version and type

Of the nearly 6 thousand nodes, 554 fail to report or were "missing" and only eleven reported as extra as shown in Figure 4.

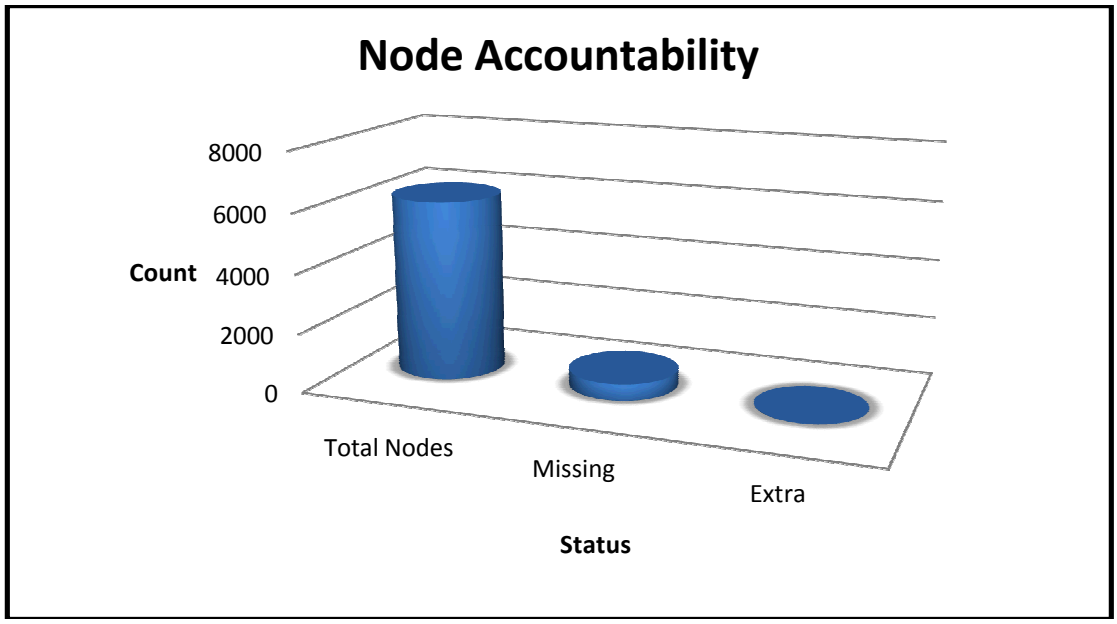

Figure 4- Node accountability as reported by Splunk

Further analysis confirmed that 95% of the nodes are running *toss-release-2.3-3.ch5.3*, 3% are running *toss-release-2.3-3rc1.ch5.3,* and 2% of the nodes are running other OS versions.

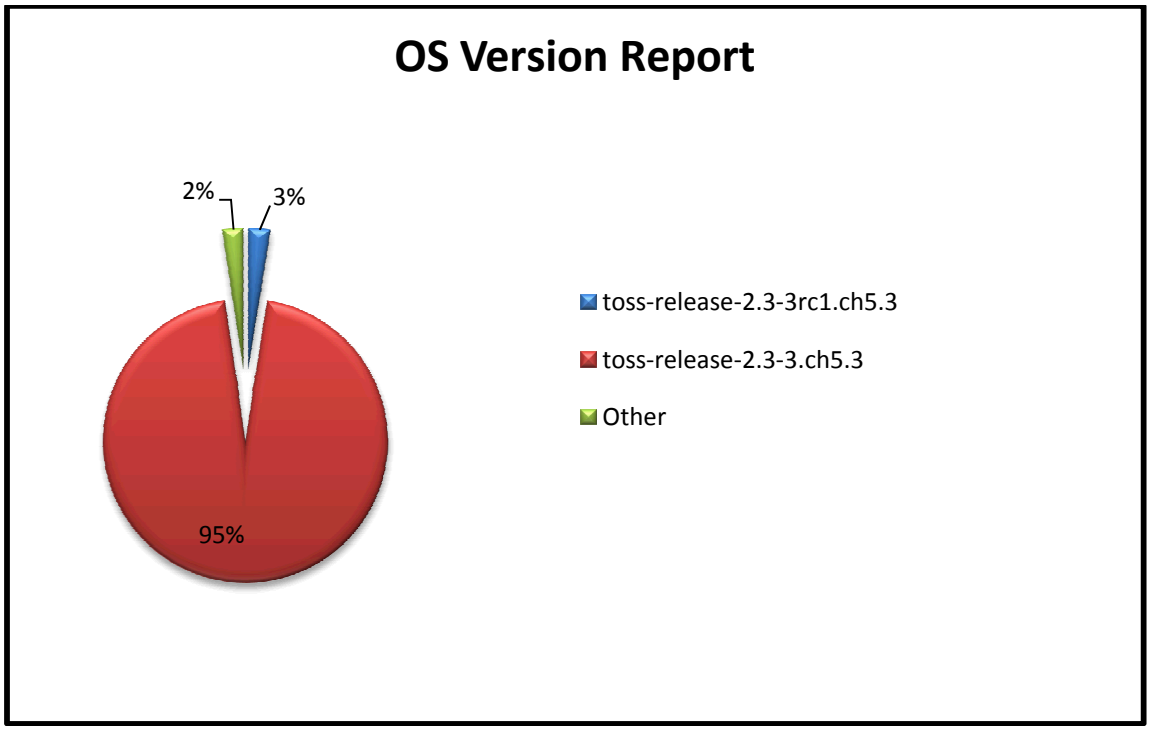

Figure 5 – Operating System Breakdown

Discussion/Conclusion

Our implementation of the continuous monitoring and reporting tool provided near real-time, detailed, and reliable information regarding configuration compliance and security patch levels. The agent was deployed on over thirty clusters totaling over six thousand nodes at the HPC center. The results obtained from these reports gave system administrators a better situational awareness of the HPC clusters. This information will also be used for auditing purposes.

Results obtained from deploying the agent revealed OS and kernel versions and types. Table 3 confirmed that *vulcanlac3, zmist1, and hype223* are configured differently. Running this report after a cluster-wide update would provide actionable information about nodes that passed and failed to get the latest release. Failure to address the issue in a timely manner could potentially render those hosts vulnerable to malicious threats. Furthermore, figure 4 gives an overview of active and inactive nodes. Nodes that are powered off or offline would be identified as "missing", while nodes that reported when they should not have, would be reported as "extra." Missing and extra nodes could potentially indicate configuration issues, hosts down, or hosts that have been compromised and their ability to report compromised. System administrators and security monitors can point and see in real time malfunctioning nodes acting swiftly to determine the root cause of the problem. Finally, Figure 5 provides an overall state of the systems showing that 95% of the nodes have been updated to the latest *toss-release-2.3-3.ch5.3*.

Further development of the tool could be extended not only to track OS updates or security compliance statuses, but to monitor other system's critical components as well. For example, an agent that determines the time when a particular node or cluster became noncompliant or the exact time it transitioned to a compliant state or an agent that collects CPU temperature and how it correlates to its throughput. The implementation of agents is left intentionally unspecified and can be tailored as needed. This means that future users of this system can implement an agent in any language or using any tool they desire, and they can still use your framework and reporting tools. The proof of concept agent was written in bash script, and utilized the logger utility, but there are lots of tools capable of doing the job depending on the needs of future users.

Acknowledgements

This research was conducted at Lawrence Livermore National Laboratory under the Science and Technology Directorate. I thank Barry Goldman for giving me the opportunity to participate in the SULI program and the chance to have a remarkable learning experience. Special thanks go to my mentors David Fox and Adam Bertsch for giving me the opportunity to work as part of their team. Christopher Lawson for his willingness to listen and help. I gained invaluable research

and programming experience during my internship. They have motivated me to continue working on advancing my problem solving and coding skills. I am grateful for their patience and understanding. I feel proud to have been part of the LLNL family and the LC group in particular.

References

[1] Bacle, A. (n.d.). White House is treating Sony hack as 'serious national security matter' Retrieved May 4, 2015, from http://www.ew.com/article/2014/12/18/white-house-sony-interview-north-korea

[2] Clark-Wendel, P. (n.d.). The Top 5 Cybersecurity Breaches of 2014. Retrieved May 4, 2015, from http://cyber-defense.sans.org/blog/2015/01/02/the-top-5-cybersecurity-breaches-of-2014

[3] SANS Institute. (2011). *Continuous Monitoring: What It Is, Why It Is Needed, and How to Use It*. Retrieved April 4, 2015, from https://www.sans.org/reading-room/whitepapers/analyst/continuous-monitoring-is-needed-35030

[4] Utilities Tutorial. (n.d.). Retrieved May 4, 2015, from http://dev.splunk.com/view/python-sdk/SP-CAAAEFC

[5] Argparse Tutorial. (n.d.). Retrieved May 4, 2015, from ttps://docs.python.org/2/howto/argparse.html

[6] Tabulate 0.7.5 : Python Package Index. (n.d.). Retrieved May 4, 2015, from https://pypi.python.org/pypi/tabulate

[7] Barney, B. (2015, February 22). Introduction to Livermore Computing Resources. Retrieved May 4, 2015, from https://computing.llnl.gov/tutorials/lc_resources/#Hardware

[8] Stallings, W., & Brown, L. (2012). *Computer security: Principles and practice* (2nd ed.). Boston, Massachusetts: Pearson.

[9] Data interpretation: Fields and field extractions. (n.d.). In *Splunk Enterprise 6.1.3: Knowledge Manager Manual*. Splunk.

[10] Carasso, D. (2012). *Exploring Splunk: Search processing language (SPL) primer and cookbook*. New York, NY: CITO Research.

[11] Splunk, Installation and Configuration Manual. (n.d.). Retrieved April 7, 2015, from http://docs.splunk.com/Documentation/PCI/2.1.1/Install/Deploymentoptions

Glossary

Agent: a bash shell script for rerecording events to sys log

Deployment Client: is a remote Splunk instance that receives configuration files from the **deployment server**.

Event: a set of values associated with a time stamp.

Forwarders (Data Collection): collect data by monitoring folders, listening on network ports, or executing scripts. They can perform parsing tasks such as filtering and anonymizing data before transmitting it securely to indexers.

Host: refers to the node where the agent is currently running.

Indexers (Parsing): are the core of the Splunk infrastructure. They process new data received from the forwarders and store them to service search requests from the Search Head.

Search: a series of commands and arguments, chained together with pipe character (|) that takes the output of one command and feeds it into the next command.

Search Head: a Splunk Enterprise instance that performs only searching, and not indexing.

Splunk: is a powerful software platform for collecting, centralizing, indexing, and analyzing machine-generated data.

TOSS: (TCE Operating System Services) is an Ubuntu-based Linux distribution.

Search Heads (Job Services): are the view engine that provides the primary user interface. They accept user requests and dispatch actionable searches to the indexers.