RESEARCH ARTICLE OPEN ACCESS

Survey on Log Analysis and Management

Sandesh Thosar [1], Abhishek Mane [2], Swapnil Raykar [3], Rahul Jain [4]
Pallavi Khude [5], Shanthi Guru [6]
Research Scholar [1], [2], [3] & [4], Assistant Professor [5] & [6]
Dr. D.Y. Patil College of Engineering, Pune
Maharashtra – India

ABSTRACT

Security is one of the biggest concerns of any organization that has an IT infrastructure. It is very important for an administrator to always know the security posture of the network and servers that they manage. One way to always know the state of an environment is through system logs. While there are hundreds or thousands of nodes that create logs on a network, most logs are hardly ever read due to the complexity and volume of the log data. This creates a problem for the administrator as logs must be reviewed, but if the whole day is spent reviewing logs, there is never any time left over to react to the problems found in the system logs. The growing size and complexity of log files has made the manual analysis of system logs by administrators prohibitive. This fact makes it important for tools and techniques that will allow some form of automation in management and analysis of system logs to be developed. This paper discusses the methods of detecting security threats in network by collecting system logs and analyzing them with MapReduce jobs.

Keywords:- Hadoop, Logs, MapReduce, Hive, Vulnerability.

I. INTRODUCTION

Modern computing systems generate huge log data. System administrators or domain experts utilize the log data to understand and optimize system behaviors. While several techniques exist to interpret logs, describing and assessing log quality remains relatively unexplored. This paper studies various modules such as Log Collection, Log Analysis and Report Generation required for the assessment of network of an organization. Large organizations use the MapReduce programming model for processing log data. MapReduce is designed to work on data stored in a distributed file system (HDFS), so number of log collection systems has been built to copy data into HDFS. These systems often unable to handle failures, with errors being handled separately by each piece of the collection, transport and processing pipeline. This paper describes different data collection system. Some of them are built on top of Hadoop, an open-source framework for storing data and running applications on distributed system, which provides scalability and robustness. It also includes flexible tools for monitoring MapReduce jobs and analyzing results.

Nowadays every enterprise, be it small or large, depends on information technology for some or most of its operations, and along with it, issues of security arises. Most of the small and medium scale enterprises (SMEs) and sometimes large enterprises are often unaware of the information security issues and hence they ignore it. The budget (if) allocated for the security purposes is generally less to get all security compliance done like penetration testing. Ignorance to the above often leads to a security issue which finally costs more in the form of data loss and recovery costs. It is always better to adopt a proactive strategy rather than a reactive one in the field of information security. This paper discusses the vulnerability assessment utilizing some free and easy to use tools for smaller modules. These tools are easy to use so that anyone with basic technical knowledge can use them, so that even a small enterprise can utilize them to generate results and take appropriate action.

Vulnerability: Vulnerability is a weakness or flaw in the application which allows an attacker to cause undesirable operations or gain unauthorized access. Presence of vulnerability poses a threat to the user of the application as it might lead to data compromise. Example: Buffer Overflow

Threat: An event or action that might prejudice security. A threat can also be described as a potential violation of security. Example: A Virus

Attack: Any action that attempts to violate the security of a system. Example: Brute Force

Exploit: A command sequence or data chunk whose aim is to take advantage of a flaw or vulnerability in an application. Example: MS 12-020 RDP exploit.

Vulnerability assessment in field of cyber security could be defined as the process of Identifying, Enumerating and Ranking the vulnerabilities present in a system or network in order to patch them. It is concerned with the security of the resource and its environment and is a proactive approach.

A single event line in a system log will generally consist of several fields of information, see Fig. 1.The most of important of which is the natural language message (free form message), which describes the event. Free form messages generated by lines in source code are a feature of almost any event log. The use of natural language in these free form messages is responsible for their lack of structure. The ability to impose structure on free form messages in system logs will therefore greatly enhance the ability to automatically analyze and manage them. Modern computing systems generate large amounts of log data. The log data describes the status of each component and records system internal operations, such as the starting and stopping of services, detection of network connections, software configuration modifications, and execution errors System administrators or domain experts utilize the log data to understand and optimize system behaviors.

As system logs are to be collected from thousands of nodes in network, it results in large amount of data. To deal with such a BIG DATA distributed framework like Apache Hadoop could be used.

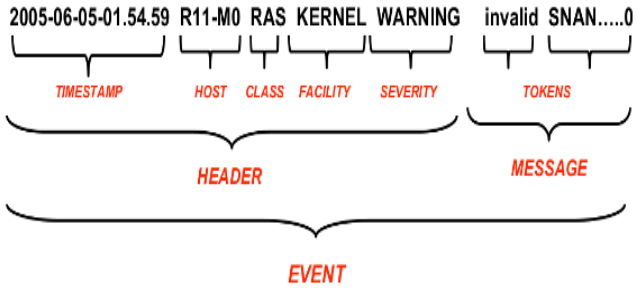

Figure 1: An Example System Log Event[19]

The technologies used are highlighted as follows:

i) Apache Hadoop:

Apache Hadoop[21] is an open source distributed framework that facilitates data intensive tasks over big data while maintaining the properties of volume, velocity and varieties. It supports the distributed application on large clusters on commodity hardware. Hadoop has two logical modules: HDFS (Hadoop Distributed File System) as storage and Map Reduced for computational workflow. HDFS provides high availability and fault tolerance by compromising on the data redundancy. Hadoop environment employs a master-slave architecture where one master node (called Job tracker) manages a number of slave nodes (called Task trackers).While storing data on HDFS, it takes random file of 64/128 Mega- bytes of blocks in size and a block is duplicated to the default replication factor of 3 in general. There are various types of ecosystems like Pig Latin, Hive, HBase and Zookeepers which are supported by Hadoop and provides added functionality to Hadoop environment. HBase supports distributed column oriented database which is built on distributed file system (HDFS).

ii) MapReduce Framework:

Hadoop MapReduce is a software frameworkfor easily writing applications which process vast amounts of data (multi-terabyte data-sets) in parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. A MapReduce job usually splits the input dataset into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and output of the job are stored in a file system. The framework takes care of scheduling tasks,monitoring them and re-executes the failed tasks. Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in every high aggregate bandwidth across the cluster.

The MapReduce framework consists of a single master Job Tracker and one slave Task Tracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master. Minimally, applications specify the input/output locations and supply map and reduce functions via implementations of appropriate interfaces and/or abstract-classes. These, and other job parameters, comprise the job configuration. The Hadoop job client then submits the job and configuration to the JobTracker which then assumes the responsibility of distributing the software configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job client.

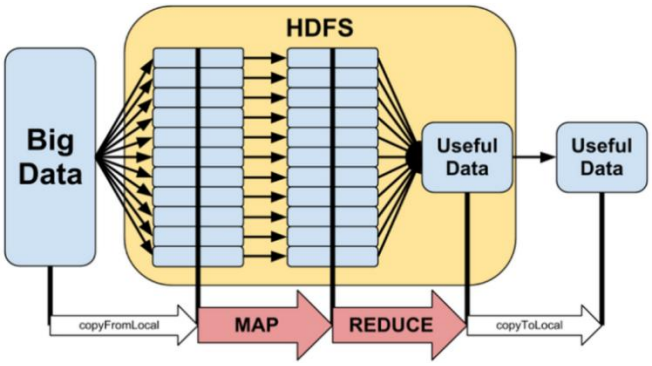

Figure 2: MapReduce Framework[20]

iii) Hive:

Apache Hive[22] is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. While initially developed by Facebook, Apache Hive is now used and developed by other companies such as Netflix. Apache Hive supports analysis of large datasets stored in Hadoop's HDFS and compatible file systems such as Amazon S3 file system. It provides an SQL like language called HiveQL with schema on read and transparently converts queries to map/reduce, Apache Tez and Spark jobs. All three execution engines can run in Hadoop YARN. To accelerate queries, it provides indexes, including bitmap indexes. By default, Hive stores metadata in an embedded Apache Derby database, and other client/server databases like MySQL can optionally be used.

While based on SQL, HiveQL does not strictly follow the full SQL-92 standard. HiveQL offers extensions not in SQL, including multi table inserts and create table as select, but only offers basic support for indexes. Also, HiveQL lacks support for transactions and materialized views, and only limited subquery support. Internally, a compiler translates HiveQL statements into a directed acyclic graph of MapReduce or Tez, or Spark jobs, which are submitted to Hadoop for execution.

II. RELATED WORK

Many existing techniques, for log collection are designed and still many are emerging for analyzing logs to detect vulnerabilities in the network or system. In this section, we list the most relevant techniques and discuss about them.

i) Syslogd:

The Unix syslogd deamon, developed in the 1980s, supported cross-network logging[5]. Robustness and fault-tolerance were not design goals. The original specification for syslogd called for data to be sent via UDP and made no provision for reliable transmission. Today, syslogd still lacks support for failure recovery, for throttling its resource consumption, or for recording metadata. Messages are limited to one kilobyte, inconveniently small for structured data.

ii) Splunk:

Splunk[6] is a commercial system for log collection, indexing and analysis. It relies on a centralized collection and storage architecture. It does not attempt high availability, or reliable delivery of log data. However, it does illustrate the demand in industry for sophisticated log analysis. To satisfy this need, many large Internet companies have built sophisticated tools for large-scale monitoring and analysis. Log analysis was one of the original motivating uses of MapReduce, and the associated Sawzall scripting language[7,8].

iii) Chukwa:

Chukwa[18] represents a design point in between two existing classes of systems: log collection frameworks on the one hand, and network management systems on the other. Chukwa intends to combine the abundance of data display tools of existing NMS systems, with the high throughput and robustness expected of log collection frameworks. Chukwa has some similarity with network monitoring systems such as Nagios, Ganglia, or Tivoli Monitoring[2,3,4]. The three systems differ in emphasis, but have important commonalities. All are capable of collecting and storing substantial volumes of metrics data. All include tools for displaying this data. Nagios and Tivoli monitoring have centralized architectures, while Ganglia is decentralized. Ganglia, unfortunately, is heavily adapted towards numeric time-series data, and provides minimal support for the sort of complex text-processing necessary for our applications. Chukwa, however, differs in crucial respects from these current systems.

Today's monitoring systems are focused primarily on collection, with storage being a secondary priority. Chukwa is designed for far higher data rates; metrics data, which is essentially all that Ganglia and Nagios are used to collect, is only a few percent of the data we will capture in operational settings. With hundreds of gigabytes of data being collected per day, processing the stored data becomes a key bottleneck. Chukwa's design was optimized precisely for storage and batch processing of collected data. While MapReduce is routinely used at these scales, no currently available monitoring system makes provision for large-scale data intensive processing.

iv) Scribe:

Scribe[1] is the one that is open source tool. Scribe is a service for forwarding and storing monitoring data. The Scribe meta data model is much simpler than that of Chukwa: messages are key-value pairs, with both key and value being arbitrary byte fields. This has the advantage of flexibility. It has the disadvantage of requiring any organization using Scribe to develop its own metadata standard, making it harder to share code between organizations. A Scribe deployment consists of one or more Scribe servers arranged in a directed acyclic graph with a policy at each node specifying whether to forward or store incoming messages. In contrast to Chukwa, Scribe is not designed to interoperate with legacy applications. The system being monitored must send its messages to Scribe via the Thrift RPC service. This has the advantage of avoiding a local disk write in the common case where messages are delivered without error. It has the disadvantage of requiring auxiliary processes to collect data from any source that hasn't been adapted to use Scribe. Collecting log files from a non-Scribe-aware service would require using an auxiliary process to tail them. In contrast, Chukwa handles this case smoothly. Scribe makes significantly weaker delivery guarantees than Chukwa. Once data has been handed to a Scribe server, that server has responsibility for the data. Any durable buffering for later delivery is the responsibility of the server, meaning that the failure of a Scribe server can cause data loss. There can be no end-to-end delivery guarantees, since the original sender does not retain a copy. Clients can be configured to try multiple servers before giving up, but if a client cannot find a working Scribe server, data will be lost.

v) Artemis:

Another related system is Artemis, developed at Microsoft Research to help debug large Dryad clusters[8]. Artemis is designed purely for a debugging context: it processes logs on the machines where they are produced, using Dryad LINQ[12] as its processing engine. The advantage of this architecture is that it avoids redundant copying of data across the network, and enables machine resources to be reused between the system being analyzed and the analysis. The disadvantage is that queries can give the wrong answer if a node crashes or becomes temporarily unavailable. Artemis was not designed to use long-term durable storage, which requires replication off-node. Analysis on-node is also a poor fit for monitoring production services. Analyzing data where it is produced risks having data analysis jobs interfere with the system being monitored. Chukwa is flexible enough to emulate Artemis if desired, in situations with large data volumes per node. Instead of writing across a network, agents could write to a local Hadoop file system process, with replication disabled. Hadoop could still be used for processing, although having only a single copy of each data item reduces the efficiency of the task scheduler[13].

vi) Flume:

Flume[2] is another, more recent system developed for getting data into HDFS. Flume was developed after Chukwa, and has many similarities: both have the same overall structure, and both do agent-side replay on error. There are some notable differences as well. In Flume, there is a central list of ongoing data flows, stored redundantly in Zookeeper. Whereas Chukwa does this end-to-end, Flume adopts a more hop-by-hop model. In Chukwa, agents on each machine are responsible for deciding what to send.

vii) Snort:

The related paper[9] is based on the distributed processing of snort alert using Hadoop framework, which handles large amount of network logs. The objective of this paper is to gather warning messages from multiple snort processes which run on different servers and analyze the logs using Hadoop MapReduce. The system uses Hadoop environment which consists of multiple slave node for better performance than a single computer system. Experimental results of system by using 8 slave nodes in a network exhibits a performance improvement which is about 4.2 times faster speed then that of single computer system. The system is scalable in nature but it lacks in real time property. In order to add real-time data processing, system requires investigating with different Hadoop ecosystems like Cloudera Impala, Scoop and Storm.

viii) NIDS:

In related paper[10] is based on the scalable NIDS log analysis using cloud computing infrastructure. The Primary objective is to efficiently handle large volume of NIDS logs from servers by using Hadoop and cloud infrastructure. Once enough data is collected then it is necessary to analyze rapidly and determine whether any attacks or malicious activity are present in a network. Performance analysis is carried out on Snort log report with file size of 4 GBytes. The execution time for log analysis is found by comparing Hadoop based system with single computer system without Hadoop. In the result, as number of nodes increase, the system performance increases as compared to the single system. When number of nodes in the system is 5, performance of the system is about 2.5 times better than that of the single system. In future improvement related to data analytic the system is open to adopt Hadoop ecosystems like Hbase, Hive with zookeeper for dynamic scheduling. This is flexible to perform auto versioning to data in HDFS.

There are also a number of more specialized monitoring systems worth mentioning. Tools like Astrolabe, Pier, and Ganglia[14,15,16] are designed to help users query distributed system monitoring data. In each case, an agent on each machine being monitored stores a certain amount of data and participates in answering queries. They are not designed to collect and store large volumes of semi-structured log data, nor do they support a general-purpose programming model. Instead, a particular data aggregation strategy is built into the system. This helps achieve scalability, at the cost of a certain amount of generality. In contrast, Chukwa separates the analysis from the collection, so that each part of a deployment can be scaled out independently.

ix) Fluentd:

Fluentd[17] is an open source data collector, which lets you unify the data collection and consumption for a better use and understanding of data. Fluentd tries to structure data as JSON as much as possible: this allows Fluentd to unify all facets of processing log data: collecting, filtering, buffering, and outputting logs across multiple sources and destinations (Unified Logging Layer).

The downstream data processing is much easier with JSON, since it has enough structure to be accessible while retaining flexible schemas. Fluentd decouples data sources from backend systems by providing a unified logging layer in between. This layer allows developers and data analysts to utilize many types of logs as they are generated. Just as importantly, it mitigates the risk of "bad data" slowing down and misinforming your organization. A unified logging layer lets you and your organization make better use of data and iterate more quickly on your software.

III. CONCLUSION

This paper described different techniques to collect and analyze logs, which are very important source of information and could be used for various security purposes such as a NetFlow, IDS detection, firewall log activity, file access, system error, or login failure.

REFERENCES

[1] Scribe. *http://sourceforge.net/scribeserver/projects*, 2008.

[2] Cloudera's flume. *http://github.com/cloudera/flume*, June 2010.

[3] M. Aharon, G. Barash, I. Cohen, and E. Mordechai.One graph is worth a thousand logs: Uncovering hidden structures in massive system event logs. In European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, Bled, Slovenia, September 2009.

[4] M. Armbrust, A. Fox, R. Griffith, A. Joseph,R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, et al. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report 2009-28, UC Berkeley, 2009.

[5] C. Lonvick. RFC 3164: The BSD syslog Protocol. *http://www.ietf.org/rfc/rfc3164.txt*, August 2001.

[6] Splunk Inc. IT Search for Log Management, Operations, Security and Compliance. *http://www.splunk.com/, 2009*.

[7] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J.DeWitt, S. Madden, and M. Stonebraker. A Comparison of Approaches to Large-Scale Data Analysis. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, Providence, RI, 2009.

[8] G. F. Cret¸u-Ciocarlie, M. Budiu, and M. Goldszmidt. Hunting for problems with Artemis. In First USENIX Workshop on Analysis of System Logs (WASL '08), San Diego, CA, December 2008.

[9] JeongJin Cheon,"Distributed Processing of Snort Alert Log using Hadoop",Department of Computer Engineering ,Kumoh National Institute of Technology ,Gumi, Gyeongbuk, Korea.

[10] Manish Kumar,"Scalable Intrusion Detection Systems Log Analysis using Cloud Computing Infrastructure", M. S. Ramaiah Institute of Technology, Bangalore and Research Scholar, Department of Computer Science and Applications, Bangalore University, Bangalore, INDIA.

[11] Prathibha,"Design of a Hybrid Intrusion Detection System using Snort and Hadoop", International Journal of Computer Applications (0975 – 8887),Volume 73– No.10, July 2013

[12] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. Gunda, and J. Currey. DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language. In 8th

USENIX Symposium on Operating Systems Design and Implementation (OSDI '08), San Diego, CA, December 2008.

[13] O. O'Malley and A. C. Murthy. Winning a 60 Second Dash with a Yellow Elephant. *http://sortbenchmark.org/Yahoo2009.pdf*, April 2009.

[14] R. Huebsch, J. Hellerstein, N. Lanham, B. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. Proceedings of 19th International Conference on Very Large Databases (VLDB), pages 321–332, 2003.

[15] R. Van Renesse, K. Birman, and W. Vogels. As trolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. ACM TOCS, 21(2):164–206, 2003

[16] M. Massie, B. Chun, and D. Culler. The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. Parallel Computing,30(7):817–840, 2004.

[17] Fluentd, *www.fluentd.org*.

[18] Rabkin Ariel and Randy H. Katz. "Chukwa: A System for Reliable Large-Scale Log Collection." *LISA*. Vol. 10. 2010.

[19] Makanju, Adetokunbo, A. Nur Zincir-Heywood, and Evangelos E. Milios. "Storage and retrieval of system log events using a structured schema based on message type transformation." *Proceedings of the 2011 ACM Symposium on Applied Computing*. ACM, 2011.

[20] MapReduce Workflow, *http://www.glennklockwood.com/data-intensive/hadoop/mapreduce-workflow.png*

[21] Apache Hadoop *https://hadoop.apache.org/*

[22] Apache Hive *https://hive.apache.org/*