ISSN 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 2, April 2012

An Evaluation of Log File and Compression Mechanism

Raj Kumar Gupta[1], Rashmi Gupta[2]

Computer Science Department,
Technocrats Institute of Technology, Bhopal (M.P.)

***Abstract*— with worldwide development of multi-national company's communication infrastructure required to increase. As the size of these computer networks increases, it becomes more and more difficult to monitor, control, and secure them. Networks consist of a number of diverse devices, sensors, and gateways which are often spread over large geographical areas. Each of these devices produces log files, which need to be analyzed and monitored to provide network security and satisfy regulations. current information systems are replete with log files, created in multiple places (e.g., network servers, database management systems, user monitoring applications, system services and utilities) for multiple purposes (e.g., maintenance, security issues, traffic analysis, legal requirements, software debugging, customer management, user interface usability studies). Log files in complex systems may quickly grow to huge sizes. Often, they must be kept for long periods of time. For reasons of convenience and storage economy, log files should be compressed. However, most of the available log file compression tools use general-purpose algorithms (e.g., Deflate) which do not take advantage of redundancy specific for log files. Presented paper describes an optimal framework for log-file compression.**

***Keywords*— Log file, Cyber Forensic, File compression**

I. INTRODUCTION

The storage requirement for operational data is increasing as increase in the computing technology. Current computer networks consist of a myriad of interconnected, heterogeneous devices in different physical locations (this is termed a distributed computing environment). Computer networks are used by many different sized corporations, educational institutions, small businesses and research organizations. In many of these networks, monitoring is required to determine the system state, improve security or gather information. Examples of such scenarios are: large global corporate networks, which consist of many different mail servers, web servers, file servers and other machines spanning different countries which need to be monitored to protect valuable information and distributed mail servers which log to a central location. In these scenarios, large quantities of information need to be gathered for analytical or archival purposes. This requires the reduction of the quantity of data to minimize bandwidth utilization, maximize throughput and reduce storage space requirements .The path with the purpose of visitors take through your pages and visitors follow within your site is clear if the log file contains an entry for every page viewed .Log files are used to record

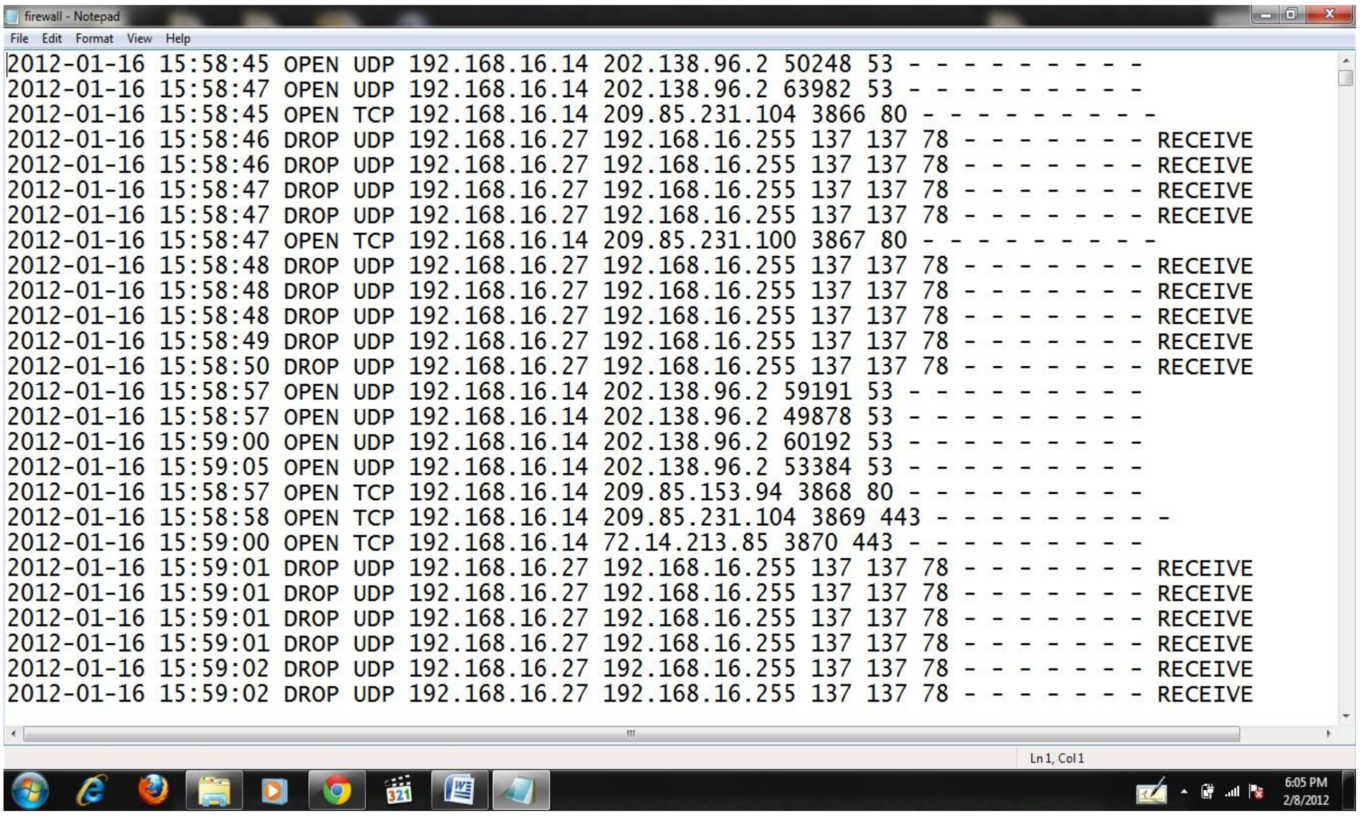
the activities that go in and out of a particular server. The log file must contain a person ID such as a login to the server or to the user's own computer. Most web sites do not require users to log in, and most web servers do not make a "back door" request to learn the user's login identity on his/her own computer. The log file does provide information about the requesting host. This information might identify a single-user computer, enabling unique identification for episode tracking. More often it is an IP address temporarily assigned by an Internet service provider (ISP) or corporate proxy server to a user's TCP/IP connection to your site, preventing unique identification. This paper is organized as follow: Section I gives the introduction of the log file as cyber crime evidence. Section II is helpful to understand the background of log file. Section III, IV, V, VI &VII explains the log file compression issue, type & technique. Section VIII is helpful to understand the background of related work. Section IX explains the proposed framework and at last section X concludes the paper and followed by the references.

II. LOG FILE

Log files are excellent sources for determining the health status of a system and are used to capture the events happened within an organization's system and networks. Logs are a collection of log entries and each entry contains information related to a specific event that has taken place within a system or network [1]. Many logs within an association contain records associated with computer security which are generated by many sources, including operating systems on servers, workstations, networking equipment and other security software's, such as antivirus software, firewalls, intrusion detection and prevention systems and many other applications. Routine log analysis is beneficial for identifying security incidents, policy violations, fraudulent activity, and operational problems. Logs are also useful for performing auditing and forensic analysis, supporting internal investigations, establishing baselines, and identifying operational trends and long-term problems [2].A log file is used to track the operation performed by any user simply by storing messages generated by an application, service, or an operating system. For example web servers maintain a log files record for every request made to the server. Log file is generally in American standard code for information interchange code file format having a .log extension. Log file is also generated by different functioning logs and alert services [3].

Initially, logs were used for troubleshooting problems, but nowadays they are used for many functions within most organizations and associations, such as optimizing system and network performance, recording the actions of users, and providing data useful for investigating malicious activity [1]. Logs have evolved to contain information related to many different types of events occurring within networks and systems. Within an organization, many logs contain records related to computer security; common examples of these computer security logs are audit logs that track user authentication attempts and security device logs that record possible attacks [6].

With the world wide deployment of network servers, service station and other computing devices, the number of threats against networks and systems have greatly increased in number, volume, and variety of computer security logs and with the revolution of computer security logs, computer security log management are required[1]. Log management is essential to ensure that computer security records are stored in sufficient detail for an appropriate period of time. Log management is the process for generating, transmitting, storing, analysing, and disposing of computer security log data. The fundamental problem with log management is effectively balancing a limited quantity of log management resources with a continuous supply of log data. Log generation and storage can be complicated by several factors, including a high number of log sources; inconsistent log content, formats, and timestamps among sources; and increasingly large volumes of log data [1, 4, and 5]. Log management also involves protecting the confidentiality, integrity, and availability of logs. Another problem with log management is ensuring that security, system, and network administrators regularly perform effective analysis of log data.

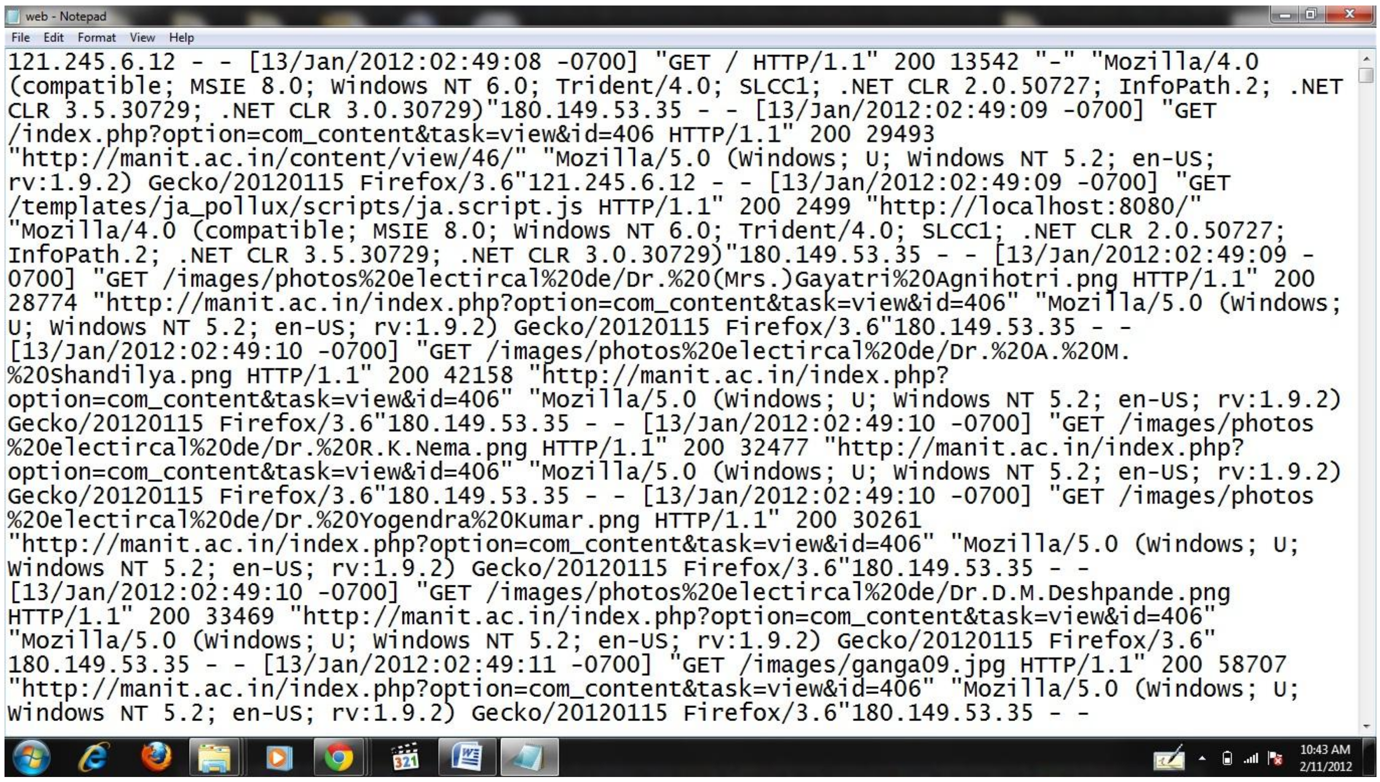


```
2012-01-16 15:58:45 OPEN UDP 192.168.16.14 202.138.96.2 50248 53 - - - - - - - - -
2012-01-16 15:58:47 OPEN UDP 192.168.16.14 202.138.96.2 63982 53 - - - - - - - - -
2012-01-16 15:58:45 OPEN TCP 192.168.16.14 209.85.231.104 3866 80 - - - - - - - - -
2012-01-16 15:58:46 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:58:46 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:58:47 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:58:47 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:58:47 OPEN TCP 192.168.16.14 209.85.231.100 3867 80 - - - - - - - - -
2012-01-16 15:58:48 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:58:48 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:58:48 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:58:49 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:58:50 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:58:57 OPEN UDP 192.168.16.14 202.138.96.2 59191 53 - - - - - - - - -
2012-01-16 15:58:57 OPEN UDP 192.168.16.14 202.138.96.2 49878 53 - - - - - - - - -
2012-01-16 15:59:00 OPEN UDP 192.168.16.14 202.138.96.2 60192 53 - - - - - - - - -
2012-01-16 15:59:05 OPEN UDP 192.168.16.14 202.138.96.2 53384 53 - - - - - - - - -
2012-01-16 15:58:57 OPEN TCP 192.168.16.14 209.85.153.94 3868 80 - - - - - - - - -
2012-01-16 15:58:58 OPEN TCP 192.168.16.14 209.85.231.104 3869 443 - - - - - - - - -
2012-01-16 15:59:00 OPEN TCP 192.168.16.14 72.14.213.85 3870 443 - - - - - - - - -
2012-01-16 15:59:01 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:59:01 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:59:01 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:59:01 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:59:02 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
2012-01-16 15:59:02 DROP UDP 192.168.16.27 192.168.16.255 137 137 78 - - - - - - - RECEIVE
```

Figure1: Example of firewall log

ISSN 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 2, April 2012



```
121.245.6.12 - - [13/Jan/2012:02:49:08 -0700] "GET / HTTP/1.1" 200 13542 "-" "Mozilla/4.0
(compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR 2.0.50727; InfoPath.2; .NET
CLR 3.5.30729; .NET CLR 3.0.30729)"180.149.53.35 - - [13/Jan/2012:02:49:09 -0700] "GET
/index.php?option=com_content&task=view&id=406 HTTP/1.1" 200 29493
"http://manit.ac.in/content/view/46/" "Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US;
rv:1.9.2) Gecko/20120115 Firefox/3.6"121.245.6.12 - - [13/Jan/2012:02:49:09 -0700] "GET
/templates/ja_pollux/scripts/ja.script.js HTTP/1.1" 200 2499 "http://localhost:8080/"
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR 2.0.50727;
InfoPath.2; .NET CLR 3.5.30729; .NET CLR 3.0.30729)"180.149.53.35 - - [13/Jan/2012:02:49:09 -
0700] "GET /images/photos%20electircal%20de/Dr.%20(Mrs.)Gayatri%20Agnihotri.png HTTP/1.1" 200
28774 "http://manit.ac.in/index.php?option=com_content&task=view&id=406" "Mozilla/5.0 (Windows;
U; Windows NT 5.2; en-US; rv:1.9.2) Gecko/20120115 Firefox/3.6"180.149.53.35 - -
[13/Jan/2012:02:49:10 -0700] "GET /images/photos%20electircal%20de/Dr.%20A.%20M.
%20Shandilya.png HTTP/1.1" 200 42158 "http://manit.ac.in/index.php?
option=com_content&task=view&id=406" "Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.9.2)
Gecko/20120115 Firefox/3.6"180.149.53.35 - - [13/Jan/2012:02:49:10 -0700] "GET /images/photos
%20electircal%20de/Dr.%20R.K.Nema.png HTTP/1.1" 200 32477 "http://manit.ac.in/index.php?
option=com_content&task=view&id=406" "Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.9.2)
Gecko/20120115 Firefox/3.6"180.149.53.35 - - [13/Jan/2012:02:49:10 -0700] "GET /images/photos
%20electircal%20de/Dr.%20Yogendra%20Kumar.png HTTP/1.1" 200 30261
"http://manit.ac.in/index.php?option=com_content&task=view&id=406" "Mozilla/5.0 (Windows; U;
Windows NT 5.2; en-US; rv:1.9.2) Gecko/20120115 Firefox/3.6"180.149.53.35 - -
[13/Jan/2012:02:49:10 -0700] "GET /images/photos%20electircal%20de/Dr.D.M.Deshpande.png
HTTP/1.1" 200 33469 "http://manit.ac.in/index.php?option=com_content&task=view&id=406"
"Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.9.2) Gecko/20120115 Firefox/3.6"
180.149.53.35 - - [13/Jan/2012:02:49:11 -0700] "GET /images/ganga09.jpg HTTP/1.1" 200 58707
"http://manit.ac.in/index.php?option=com_content&task=view&id=406" "Mozilla/5.0 (Windows; U;
Windows NT 5.2; en-US; rv:1.9.2) Gecko/20120115 Firefox/3.6"180.149.53.35 - -
```

Figure2: **Example of web log**

III. LOG FILE COMPRESSION

This section motivates the need for data reduction and explains how data compression can be used to reduce the quantity of data. It also discusses the need for the approaches which can be used and the role of log files. In many environments, tracked logs can happen very often. As a result, there is a huge amount of data produced this way every day. And often it is necessary to store them for a long period of time. Regardless of the type of recorded logs, for reasons of simplicity and convenience, they are usually stored in plain text log files. Both the content type and the storage format suggest that it is possible to significantly reduce the size of log files through lossless data compression, especially if specialized algorithm was used. The smaller, compressed files have the advantages of being easier to handle and saving storage space.

IV. TYPES OF COMPRESSION

Lossless compression algorithms frequently use statistical redundancy in such a way as to represent the sender's data more concisely without error. Lossless compression is possible as most real-world data has statistical redundancy [10]. For instance, in English text, the letter 'e' is much more used than the letter 'z', and the probability that the letter 'q' will be followed by the letter 'z' is very tiny. Another kind of compression called lossy data compression or perceptual coding is possible in this scheme some loss of data is acceptable. In general a lossy data compression will be guided by research on how people perceive the data in question. For example, the human eye is more sensitive to slight variations in luminance than it is to variations in color. JPEG image compression works in part by "rounding off" some of this less-important information. Lossy data compression provides a way to obtain the best fidelity for a given amount of compression.

Lossless compression schemes are reversible so that the original data can be reconstructed, while lossy schemes accept some loss of data in order to achieve higher

ISSN 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 2, April 2012

compression. However, lossless data compression algorithms will always fail to compress some files; indeed, any compression algorithm will necessarily fail to compress any data containing no discernible patterns. Attempts to compress data that has been compressed already will therefore usually result in an expansion, as will attempts to compress all but the most trivially encrypted data.
In practice, lossy data compression will also come to a point where compressing again does not work, although an extremely lossy algorithm, like for example always removing the last byte of a file, will always compress a file up to the point where it is empty.

V. LOSSY COMPRESSION

Lossy image compression is exercised in digital cameras, to increase storage capacity with negligible degradation of picture quality. Similarly, DVDs use the lossy MPEG-2 Video codec for compression. Compression of individual speech is often performed with more specialized methods, so that "speech compression" or "voice coding" is sometimes distinguished as a separate discipline from "audio compression". Different audio and speech compression standards are listed under audio codes. Voice compression is used in Internet telephony, for example while audio compression is used for CD ripping and is decoded by audio players.

VI. LOSSLESS COMPRESSION

The Lempel–Ziv (LZ) compression methods are the most popular for lossless storage. DEFLATE is a variation of LZ which is optimized for decompression speed and compression ratio [11], therefore compression can be slow. DEFLATE is used in PKZIP, gzip and PNG. LZW (Le-mpel–Ziv–Welch) is used in GIF images. Also worth mentioning are the LZR (LZ–Renau) methods, which serve as the basis of the Zip method. LZ methods utilize a table-based compression model where table entries are substituted for repeated strings of data. For most LZ methods, this table is generated dynamically from earlier data in the input. The table itself is often Huffman encoded (e.g. SHRI, LZX). A current LZ-based coding scheme that performs well is LZX, used in Microsoft's CAB format. The very best modern lossless compressors use probabilistic models, such as prediction by partial matching. The Burrows–Wheeler transform can also be viewed as an indirect form of statistical modelling.

VII. COMPRESSION TECHNIQUES

Compression programs exploit the redundancy in files to create smaller files which can be decompressed to produce the original file. There are many different types of compressors. They may be block-based, dividing the data into blocks and running an algorithm on each block; dictionary based, where they replace words with references to a dictionary; or they may use complex statistical models to predict future characters (statistically-based compressors) be performed to improve the possibility of compression. These may be transformations (such as Burrows-Wheeler transform) or normalizations (such as Branch Call Jump (BCJ), where jump targets in executable files are normalized before compression) of data. Compression may also be performed using the distribution of characters in the data to create statistical codes (such as Arithmetic and Range coders), new fixed length codes or variable length codes (usually prefix free codes such as Huffman coding). Compressors often use adaptive algorithms where the statistics used are updated by the encoders and decoders as they encode the data source and decode the encoded data respectively.

VIII. RELATED WORK

The Comprehensive Log Compression (CLC) method provides a powerful tool for any analysis that inspects data with lot of redundancy. Only very little a priori knowledge is needed to perform the analysis. The method provides a mechanism to separate different information types from each other. The CLC method identifies frequent repetitive patterns from a log database and can be used to emphasize either the normal course of actions or exceptional log entries or events in the normal course of actions. This is especially useful in getting knowledge out of previously unknown domains or in analyzing logs that are used to record unstructured and unclassified information[7].Szymon Grabowski, Sebastian Deorowicz Presented a specialized lossless Apache web log pre-processor [8] and test it with combination of several popular general-purpose compressors. The test results show the proposed transform improves the compression efficiency of general-purpose compressors on average by 65% in case of gzip and 52% in case of bzip2.they presented two relatively simple off-line pre-processing schemes for web log compression. Meng-Hang Ho, Hsu-Chun Yen design and implement a dictionary-based compressed pattern matching algorithm [9].Takes advantage of the dictionary structure common in the LZ78 family. With the help of a slightly modified dictionary structure, we are able to do 'block decompression' (a key in many existing compressed pattern matching schemes) as well as pattern matching on-the-fly, resulting in performance improvement as our experimental results indicate. On other hand The Lempel-Ziv-Welch (LZW) compression algorithm is widely used because it achieves an excellent compromise between compression performance and speed of execution. A simple way to improve the compression without significantly degrading its speed is proposed, and experimental data shows that it works in practice. Even better results are achieved with an additional optimization of "phasing in" binary numbers. The better compression is achieved by a coding technique, the harder it becomes to extract each percent of additional compression. It has not been easy to find easily implementable methods for improving the performance of LZC, especially when they impose an additional constraint that the execution time requirements should not be severely affected. We have selected two ways of improving LZC (the UNIX compress command). One, a method of loading the dictionary at a faster rate, has not been used before. The other, a method to phase in increased lengths of binary numbers gradually, is not original but is not currently used with LZC. Together, these two compression methods achieve substantial improvements, especially on shorter files where the dictionary does not normally have a chance to fill to an extent that achieves good compression performance.

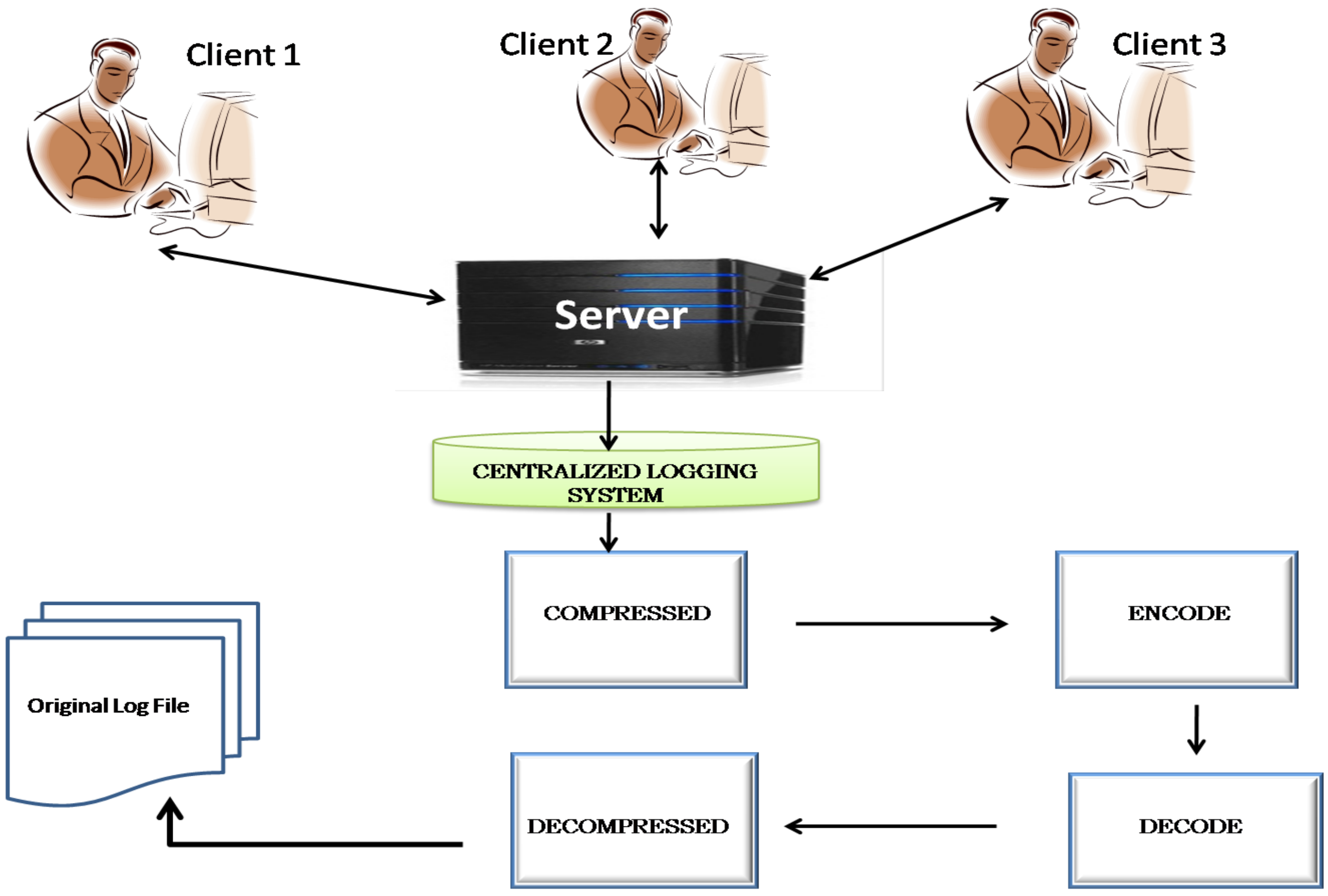


Figure3: Proposed Framework for Compression

IX. Proposed Framework

A multi-tiered log file compression solution shall be proposed. Every of the three tiers addresses one notion of redundancy (as shown in figure:3).The first tier handles the resemblance between neighboring lines. The second tier handles the global repetitiveness of tokens and token formats. The third tier is general-purpose compressor which handles all the redundancy left after the previous stages. The tiers are not only optional,

ISSN 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 2, April 2012

but each of them is designed in several variants differing in required processing time and obtained compression ratio. This way user with different requirements can find combinations which suit them best. We propose five processing schemes for reasonable ratios of compression time to log file size reduction. A collection of scripts to determine the compression ratios, compression times and decompression times when using data compression was compiled. These scripts were used to run tests on a collection of log files and the obtained statistics recorded.

X. Conclusion

Current information systems are full up with log files, often taking a considerable amount of storage space. It is reasonable to have them compressed, yet the general purpose algorithms do not take full advantage of log files redundancy. The existing specialized log compression schemes are either focused on very narrow applications, or require a lot of human assistance making them impractical for the general use. We have suggested and will try for a fully reversible log file transform capable of significantly reducing the amount of space required to store the compressed log. The transform will be presented in five variants aimed at a wide range of possible applications, starting from a fast variant for on-line compression of current logs (allowing incremental extension of the compressed file) to a highly effective variant for off-line compression of archival logs.

References

[1] Nikhil Kumar Singh, Deepak Singh Tomar, Bhola Nath Roy, "*An approach tounderstand the end user behavior through log analysis*" International Journal of Computer Applications (0975 – 8887), August 2010.

[2] Karen Kent and Murugiah Souppaya, "*Guide to Computer Security Log Management*", Computer Security Division Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, 2006

[3] Muhammad Kamran Ahmed, Mukhtar Hussain and Asad Raza "*An Automated User Transparent Approach to log Web URLs for Forensic Analysis*" Fifth International Conference on IT Security Incident Management and IT Forensics 2009.

[4] Carrier, B.D., Spafford, E.H "*Defining Digital Crime Scene Event Reconstruction*" Journal of Forensic Sciences, 49(6). Paper ID JFS2004127,2004

[5] Stephenson. P, "*Application Of Formal Methods To Root Cause Analysis of Digital Incidents*", International Journal of Digital Evidence, 3(1) ,2004

[6] P. K. Sahoo ,Dr. R. K. Chottaray, "The Role of Audit Logs in Cyber Security" International Journal of Science and Advanced Technology (ISSN 2221-8386) Volume 1 No 7 September 2011

[7] Kimmo H¨at¨onen, Jean Fran¸cois Boulicaut, Mika Klemettinen, Markus Miettinen , and Cyrille Masson, "Comprehensive Log Compression with Frequent Patterns" , pp. 360–370, 2003. Springer-Verlag Berlin Heidelberg 2003

[8] Szymon Grabowski, Sebastian Deorowicz, "Web Log Compression" Institute of Computer Science, Silesian Technical University, Gliwice, Poland

[9] Meng-Hang Ho , Hsu-Chun Yen , "A Dictionary-based Compressed Pattern Matching Algorithm", Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, Republic of China.

[10] S. Read-Miller and R. A. Rosenthal. Best Practices for building a SecurityOperations Center. Computer Associates White Paper, April 2005.

[11] J. Babbin, D. Kleiman, E. Carter Jr.and J. Faircloth, and M. Burnett. "Security Log Management.Syngress" Rockland, Knox County, Maine, USA, 2006.