International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 3, Issue 8, August 2014 ISSN 2319 - 4847

A Novel Approach for Efficient Log Management using Event Filtering

Treesa Antony[1] and Lisha Varghese[2]

[1]PG Scholar in CSE, AJCE, Kanjirappally, Kerala

[2]Asst. Prof., Dept. of CSE, AJCE, Kanjirappally, Kerala

ABSTRACT

The proper functioning of any organization greatly depends on the efficient maintenance of their log records. Some of the major benefits regarding log files are they help to identify policy violations, malicious activities, tune system performance and so on. The thesis deals with the challenges involved in a logging server-based log management. Also the log files usually include duplicate entries and several standard informational entries. This contributes a major challenge to log analysts, since they do not provide any useful information to their analysis. The process of event filtering helps to efficiently resolve this problem.

Keywords: logging, event filtering, outsourcing, integrity

1. INTRODUCTION

A log is a collection of the events occurring within an organizations systems and networks [7]. Logs are composed of log entries; each entry contains information related to a specific event that has occurred within a system and/or network. Many logs within an organization contain records related to computer security. Such computer security logs are generated by many sources, like security software, such as firewalls, antivirus software and intrusion detection and prevention systems; operating systems on workstations, servers, and networking equipment and applications. The volume, count, and variety of computer security logs have increased rapidly. This has required the need for computer security log management. Log management is mainly concerned with generating, transmitting, storing, maintaining, analyzing, and disposing of log data related to computer security. Log management is essential to ensuring that computer security records are stored in sufficient detail for an appropriate time interval. Routine analysis of log is beneficial for identifying security related issues, policy violations, malicious activity, and other related problems. Log management is an approach which dealing with large volumes of computer- generated log messages. Such computer-generated log messages are otherwise called audit records, audit trails, or event-logs. Log management deals with a variety of functions. Out of them some important ones include log collection, aggregation, long-term retention, log analysis. Log management is controlled by factors like security, system and network operations and other regulatory aspects. While analyzing large amount of logs we need to face many challenges such as large log-volumes (reaching hundreds of gigabytes of data per day as per the organization structure), differences between several logs in their formats, undocumented proprietary log-formats (that resist analysis) as well as the presence of false log records in certain types of logs. Logs were originally used for troubleshooting problems. But today, they can serve a variety of purposes like recording user activities, optimizing performance, investigating malicious activities and so on. Most of these contain records related to computer security, like audit logs which eventually track authentication attempts by users and records possible attacks efficiently.

Following are basic categories of computer security logs:

1.1. Security Software Logs

Security Software Logs generally contain security related information of computers. For this function, several types of network-based and host-based security software are used. They help to protect systems as well as data, detect malicious activity and support incident response attempts. All these factors are quite important for any organizations. The most common varieties include vulnerability management software, anti-malware software, remote access software, authentication servers and intrusion detection-prevention systems. Vulnerability management software, includes patch management software and vulnerability assessment software, generally logs the patch installation history and vulnerability status of each host. This includes known vulnerabilities and missing software updates. They may also record additional information about configurations. Vulnerability management software typically runs occasionally and is likely to generate large batches of log data.

1.2. Operating System Logs and Application Logs

Operating System Logs are generated by the operating systems for servers work- stations and other networking devices like routers and switches. They usually log security related information of huge amount. System events and audit records are the common types of OS data, as far as security is concerned. The various operational actions performed by OS components are known as System Events. These include actions such as shutting down the system or starting a service. Audit records typically contain security event information. The most common examples include events such as successful and failed authentication attempts, file accesses, account changes, security policy changes, and use of privileges.OS logs may also contain information from security software and other applications running on the system. OS logs are most beneficial for identifying or investigating suspicious activity involving a particular host. This is just because of the reason that after suspicious activity is identified by security software, OS logs are often visited to get more information on the activity. Applications are commonly used to store, access, manipulate the log data used for the organizations business processes. Most generally, operating systems and security software [8] provide the foundation and protection for applications. There are two major varieties of applications. First are commercial off-the-shelf (COTS) applications. Most organizations rely on a variety of commercial off-the-shelf applications. These include e-mail servers and clients, Web browsers and servers, file servers and file sharing clients, and database servers and clients. The second is government off-the-shelf (GOTS) business applications. This is also widely accepted by many organizations.

2. RELATED WORK

Most of the organizations suffer from a set of issues when dealing with log management. One of the major issues to be considered is how to effectively balance the quantity of resources essential for log management with the supply of log data. Another factor to be considered is the log generation and storage [9]. Here also we need to consider a set of factors like inconsistency of log contents, formats among sources, number of log sources and the volume of log data. The confidentiality, integrity, and availability of logs are considered to be maintained at all times. Security is yet another factor to be concerned. This helps the analysts to perform their function effectively. [1] considers the secure maintenance of log records for extended time periods. It tries to preserve correctness, confidentiality, tamper resistance, verifiability and privacy of log records. This work succeeded in keeping these constraints to some extent. However the integrity of log records is still considered to be a problem. Also the presence of duplicate entries and informational entries increases the log size and constitutes the major disadvantage. The present architecture includes log generator, logging client, log monitor and the logging server. The logging client receives its input directly from the log generators without any prior filtering. This is main reason behind the existence of redundant entities and unnecessary entities. This makes the log file to be of huge size and also makes it to include unwanted entries. The input mentioned is the log information from various applications or organizations. These entities are supposed to be the log generators in this context. Syslog[2] is a standard for computer message logging. It permits the separation of software that generates messages from system and software that reports and analyses them. Syslog can be used for system management and security auditing as well as generalized informational, analysis and debugging messages. Also it integrates log data from different types of systems into a central repository. Since Syslog is UDP-based, it is unreliable. Also it involves possibilities for the messages to be intercepted or discarded or dropped through network congestion. Also it does not ensure the ordered delivery of packets. The Syslog protocol is a client/server protocol, which enables a logging application to transmit a maximum of 1024 bytes. Messages are sent via UDP or TCP. SSL wrapper may be used to provide a layer of encryption through SSL/TLS. Syslog-ng[3] is an open-source implementation of Syslog protocol for Unix and Unix-like systems. It enhances the original Syslog model with features like content-based filtering, flexible configuration options, ISO 8601 timestamps, TLS encryption and reliable transport using TCP. One of the major features of Syslog-ng is its ability to format messages using UNIX-shell like variable expansion. Also it has the capability to control the message flow control in network transport. Since logging is directly into the log database, it is more effective. Shamir's secret sharing [4] divides data D into n pieces in such a way that D is easily re constructible from any k pieces, but even complete knowledge of k - 1 piece reveals absolutely no knowledge about D. This technique helps in the construction of robust key management schemes for cryptographic systems that can function securely and reliably even when misfortunes destroy half the pieces and security breaches expose all but one of the remaining pieces. This scheme is said to be the (k,n) threshold scheme. The major application of this scheme is the efficient storage of cryptographic keys. If the computer on which the key is stored crashes, the key will be inaccessible. This problem is well resolved with (k,n) threshold scheme. The attractive feature of the proactive secret sharing scheme [5] is that it periodically renews its shares. Hence any adversary tries to get the key needs to break all k locations simultaneously. It involves mechanisms to detect maliciously corrupted shares and to secretly recover the correct shares even if modified.

Following are the core properties:

- Lifetime of the secret is divided into periods of time.
- At the beginning of each time period, share holders engage in an update protocol either to learn or to destruct the secret.

Frequent pattern mining in web log data [6] aims to obtain information about the navigational behavior of users. It uses 3 pattern mining approaches like Page Sets, Page Sequences and Page Graphs for discovering different types of patterns in log database. There are three steps associated with this mining: preprocessing, discovering patterns and pattern analysis. Out of these the preprocessing step provides three types of output data that used for later analysis. The frequent pattern discovery step collects the log for analysis, omits the duplicate entries and then orders them in a predefined order.

3 DELEGATING LOG MANAGEMENT

3.1 Architecture

The components of the system include log generators, logging client, logging server and log monitor.

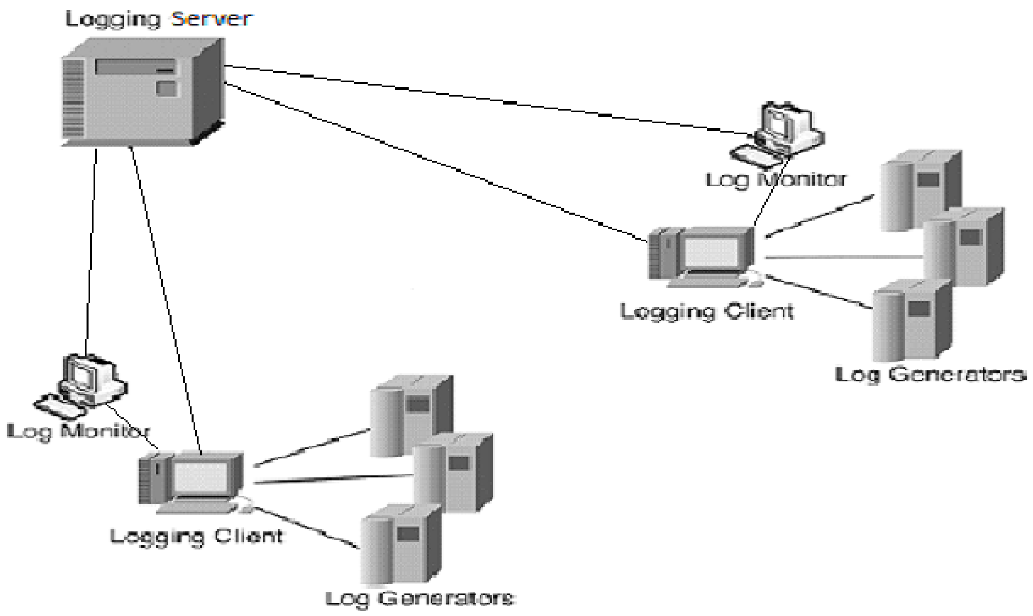

Figure 1 System Architecture for Logging Server-based secure logging

3.2 Event Filtering

The proposed method uses a logging server for the purpose of storing log data. The existing system is greatly affected with duplicate entries, as it constitutes huge amount of time to prepare them as batches. As a result, it increases both the upload time and thereafter it affects the decryption time. Event filtering [10] is the suppression of log entries from analysis, reporting, or long-term storage because their characteristics indicate that they are unlikely to contain information of interest. For example, duplicate entries and standard informational entries might be filtered because they do not provide useful information to log analysts. Typically, filtering does not affect the generation or short-term storage of events because it does not alter the original log files. It uses 3 pattern mining approaches like Page Sets, Page Sequences and Page Graphs for discovering different types of patterns in log database. There are three steps associated with this mining: pre-processing, pattern discovery and pattern analysis. Out of these the pre-processing step provides three types of output data that used for later analysis. The frequent pattern discovery step collects the log for analysis, omits the duplicate entries and then orders them in a predefined order. Following we have the diagrammatic representation of log mining

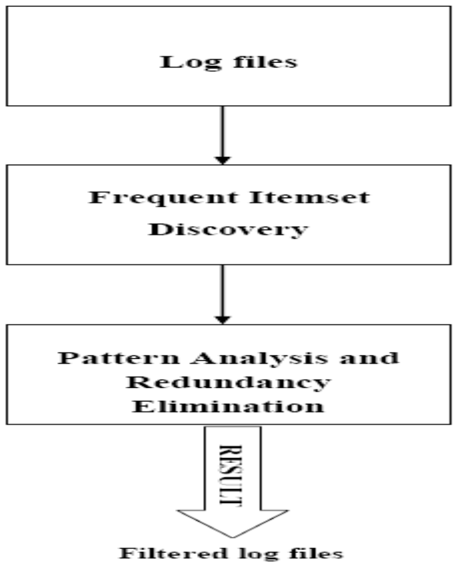
Log files
Frequent Itemset Discovery
Pattern Analysis and Redundancy Elimination
RESULT
Filtered log files

Figure 2 Log Mining Process

4 IMPLEMENTATION

4.1 Log File Preparation

The primary factor behind the log file preparation protocol is the creation and management of three sets of keys A_i and X_i which are keys for ensuring log record integrity, and K_i which is a key for ensuring log record confidentiality. These keys are derived in sequentially. The process starts with three master keys A_0, X_0, and K_0. The rest of the keys are derived by hashing each of them. Initially, the logging client randomly generates three master keys A_0, X_0, and K_0. Out of these, A_0, andX_0 are intended for ensuring log integrity, and K_0 for ensuring log confidentiality. It is necessary that these keys need to satisfy the requirements of the discussed proactive secret-sharing scheme. It then proceeds by preparing the log records. At the logging client, the log data arrives as a series of messages L_1, L_2. . . L_n. Each L_i contains a group of log records generated by a log generator. The logging client is responsible to upload prepared log records in batches of n. The value n is determined randomly at the beginning, during the time of each log batch preparation. 1. The logging client initially creates a special first log entry. This includes $L_0 = <TS, \text{log-initialization}, n>$. This is done before any log data arrives at the logging client. It then encrypts this log entry with the key K_0 and computes the message authentication code $MAC_0 = H_{A0} [E_{K0} [L_0]]$ for the encrypted entry. This is done using the key A_0. This constitutes the resulting first log entry. The client now adds it for the current batch $< E_{K0} [L_0], MAC_0 >$ to the log file. 2. The logging client then calculates new set of keys. This is computed by hashing the existing values $A_1 = H[A_0]$; $X_1 = H[X_0]$; $K_1 = H[K_0]$, Then, the logging client securely erases the previous set of keys and waits to get the next log message. 3. As soon as the first log message L_1 arrives, the logging client creates a record M_1 such that $M_1 = L_1 \parallel H_{A0} [E_{K0} [L_0]]$. It encrypts M_1 with the key K_1, and creates the corresponding message authentication code for the resulting log data. The MAC value is computed as $MAC_1 = H_{A1} [E_{K1} [M_1]]$. It also computes an aggregated message authentication code using the key X_1 as $MAC`_1 = H^n_{X1} [MAC_0 \parallel MAC_1 \parallel n]$. The new log batch entry is $< [E_{K1} [M_1], MAC_1 >$. It then creates the next set of keys as $A_2 = H[A_1]$;$X_2 = H[X_1]$;$K_2 = H[K_1]$ and securely deletes the previous keys A_1 and K_1. 4. For every new log data L_i, the process is repeated. Each time the logging client creates log file entries $< E_{Ki} [M_i];MAC_i >$. It also creates the aggregated message authentication code $MAC`_i = H^{n-i+1}_{Xi} [MAC_{i-1} \parallel MAC_i] \parallel n - i + 1]$. Once $MAC`_i$ has been generated, $MAC`_{i-1}$ is securely deleted. The logging client then creates new keys. This is done as above like $A_{i+1} = H[A_1]$,$X_{i+1} = H[X_1]$ and $K_{i+1} = H[K_1]$ and securely erases the old keys A_i, X_i and K_i. 5. Finally the client creates the last log entry M_n for the current batch from the last log data L_n, Then it creates a special log close entry LC as $LC = <E_{Kn+1}[TS; \text{log-close} \parallel MAC_n], H_{An+1}[E_{Kn+1}[TS, \text{log-close} \parallel MAC_n]] >$. Same as above the client generates an aggregated message authentication code for the special log close entry LC. It then securely erases the three keys used in this final step and uploads the resulting log batch and aggregated message authentication code to the logging server as one unit.

4.2 Upload Tag Generation

Each uploaded log batch of log records needs to be indexed by a unique key value. This indexing helps to effectively retrieve it later. Also it is to be ensured that this unique key value cannot be traced back to the logging client or to the log monitor. The logging client uploads log data, and the log monitor seeks the data. For this purpose, the log data stored at the logging server is indexed by an anonymously generated upload-tag. This upload-tag is created by the logging client together with the log monitor. However, it has the property that a given upload-tag cannot be linked either to the corresponding logging client or a log monitor [7]. An upload-tag is generally an instance of a hashed Diffie-Hellman key.

The upload-tag generation process proceeds as follows:

1) Both the logging client and the log monitor agree on a finite group G having a generating element $\alpha \epsilon G$. They choose a large prime number p. P is selected such that $p >> \alpha$ and α is a primitive root of p. These values can be chosen from publicly available information. So that all adversaries know that the logging client and the log monitor have chosen these specific values for α and p.
2) Initially the logging client, A, generates a random number r_A and keeps it as secret. The log monitor, B, also chooses independently another random number r_B. This is also kept as secret. A creates its half of the upload-tag as $T_A = Sig_A[\alpha_{rA \text{ mod} P}]$. It then uploads a message on to the logging server. So that the message includes the following elements: ID(A), TS, N, $E^{pub}_B [k_1]$, $E_{k1} [T_A] \parallel E_{k1} [m]$. This value is considered as one half of an anchor upload-tag that will be generated. The value m is used to indicate the number of times the logging client plans to use the anchor for generating upload-tags. It simply acts as a counter. Similar is the function performed by log monitor. It also generates one half of the anchor upload-tag as $T_B = Sig_B[\alpha_{rB \text{ mod} P}]$ and uploads the message ID(B), TS, N, $E^{pub}_A[k_2]$, $E_{k2}[T_B]$.
3) At some particular point of time, both A and B retrieve the other party's anchor upload-tag portion. Now each of them is able to independently compute the Diffie-Hellman key α^{rArB}mod p.
4) Thus in order to create the ith upload-tag, the logging client hashes the Diffie Hellman key (m-i) times. That is, $\text{upload-tag}_i = H^{n-i} [\alpha^{rArB} \text{mod } p]$. In this phase, the log monitor does not have to interact with the logging client . Even if, just the public parameters are given, no one can create the Diffie Hellman key α^{rArB}mod p without the secret values r_A or r_B. This is considered as a major property of the key.

4.3 Upload and Retrieval of Log Records

The data upload protocol proceeds as follows:

1. The logging client initially creates a unique upload-tag as discussed earlier. This is done for every log data that needs to be uploaded to the logging server. It also creates a unique DeleteTag as DeleteTag = H^n[randk(.)] using a cryptographically strong random number.
2. Then the logging client sends the LogData to the logging server as a message UploadTag, TS, DT, N, LogData.

4.4 Deletion of Log Records

Log deletion is an operation that is normally requested by a logging client and it is performed by the logging server. The logging server has the DeleteTag in its database. The logging client requests for a deletion via log monitor. This is because log monitor is designed as an entity that can perform operations like review, retrieve or monitor log data. The log monitor generates a delete request as per the logging client's wish. The logging client the sends (n-1) times hashed DelNum to the logging server. Upon its reception, the logging server hashes it once more to have the respective DeleteTag. Then, it checks whether the received value matches with the stored value. If yes, logging server performs deletion and updates database. The log rotate or log delete is an operation that can be requested only by authorized entities. Evidence of possession of the DeleteTag is assumed to be a proof of the necessary authorization. If an entity other than the logging client initiates a log delete request, it need to get the DeleteTag from the logging client. Also it needs to acquire the relevant Upload-tag. The requestor can also then recreate the DeleteTag.

The steps for deletion (rotation) are then as follows:

1. The requester sends a delete or rotate request to the logging server. The request message is meant for log data that satisfies Tag = Upload-tag $\wedge$ DeleteTag = DeleteTag
2. The logging server will ask the entity to prove that it is authorized to delete.
3. The entity provides the proof of authorization as $H^{n-1}[rand_k(.)]$.
4. The logging server verifies that DeleteTag = $H[H^{n-1}[rand_k(.)]]$. If verified, logging serves takes the necessary action and responds accordingly.

4.5 Event Filtering

It uses 3 pattern mining approaches like Page Sets, Page Sequences and Page Graphs for discovering different types of patterns in log database. There are three steps associated with this mining: pre-processing, pattern discovery and pat- tern analysis. Out of these the pre-processing step provides three types of output data that used for later analysis. The frequent pattern discovery step collects the log for analysis, omits the duplicate entries and then orders them in a predefined order.

Pre-processing

This is the first step of Event Filtering. Pre-processing is an important step during any of the complicated and renowned processes. This is because it helps to make other steps some more easier. Also it decreases the complexities associated with all further steps. In this context also pre-processing is necessary. Here pre-processing refers to the pre-processing of data. With our system, we are considering log data. During this step, initially it reads the log records from the log storage area. Generally this is from the default location within our system. There are two ways by which log files are being accessed. First is the non-optimized way. During this, log files are accessed as such, without prior modification or change being applied.

Pattern Discovery

During this step, it mines the log data to get identical patterns. For this process, the ItemSetCode algorithm is used. This process is otherwise known as log mining. In this work, time and source are the two factors that have been considered. The algorithm checks for patterns whose time and source are identical. It then collects all such patterns together. Thus the pattern discovery step is finished. Out of the several log entries, we are selecting only time and source. The reason is that these are the two main factors to be considered. Generally, events are identified based on the sources from which they have been generated. Sometimes, the same source may generate multiple log records simultaneously. These solely increase the number of duplicate entries and we need to remove them efficiently. Entries generated by different sources are treated differently. They are kept as such. That is how we filter log records.

Pattern Analysis and redundancy elimination

The result obtained from the second step, pattern discovery becomes the input here. Pattern discovery differentiates the log records from different sources and give it for pattern analysis. The algorithm checks for patterns whose time and source are identical. It then collects all such patterns together. Thus the pattern discovery step is finished. Out of the several log entries, we are selecting only time and source. The reason is that these are the two main factors to be considered. During pattern analysis, it collects such identical patterns obtained. Then it removes such patterns, keeping a single copy of them. This is the step of redundancy elimination. This process is performed for the entire log records. This is also continuously checked within particular time periods. The output of pattern analysis and redundancy elimination is filtered log files of decreased size. This is taken for log batch preparation by the logging client. The resultant log file is of reduced size. Hence, It takes less time for batch preparation. Similar is the case with upload and decryption time also.

5 RESULTS AND ANALYSIS

The proposed system is developed and implemented as an effective log outsourcing solution. There are three major entities within the system. First is the Logging Client, which is responsible for collecting the log data from different sources, preparing log batches, generating upload-tag for log batch upload and uploading log batch. The sources of log records refer to any application or organization that generates the log data during its operation. Second is the Log Monitor, which is able to retrieve, review and monitor log data. Each logging client has an associated log monitor. Upon determined by the logging client, the log monitor can retrieve the log information, decrypt it and monitor them. During monitoring, the log monitor computes the MAC value and then compares it with the stored MAC. It both are found to be the same, the data is assumed to be unchanged. Otherwise, it is confirmed that the integrity is disclosed. Third is the Logging Server. It provides a long-term storage and maintenance service to log data. Once stored within logging server, only the logging client can send a request to delete this information from the logging server. This process is of high security. This process involves several steps. First the log monitor request for deletion. For this, it sends a request to Logging Client to get the DeleteTag. The Logging Client sends the (n-1) times hashed DelNum to the Logging Server. The Logging Server again hashes it, and compares the result with the stored DeleteTag value. If both are same, then deletion is performed. The Logging Client, Log Monitor and the Logging Server have been developed in Java. The proposed system access the log files of the system upon which it is running from its default location. We can run a number of logging client instances upon different machines. Each logging client has its log monitor running along with it, upon the same machine. The Logging Client, upon running reads the log file from its location. These are now prepared as batches by the logging client. The method for preparing log batches is well described in module description section. It creates a java object for a log batch and uploads the batch in a special packet as soon as the batch is filled. There are several operations like send batch, retrieve batch, delete batch and so on. The protocol at the Logging client side starts by generating an initial set of keys as discussed earlier. The batch is now uploaded to logging server entity with appropriate upload and delete tags have been generated. The Log Monitor is also generated as a java application. It generally runs within the same machine upon which the logging client runs. It performs batch retrieval and delete operations. The Logging Server is considered to be the most important entity within the proposed system. Here a MySQL database engine is used to store log data. It is accompanied with the construction of small batch table schema with a few columns. This has the advantage of extremely fast MySQL tuple fetching capabilities. This uses JDBC connection to store the log data in remote MySQL database. This is done as byte representation of java objects. The batches are stored in the database with upload tag as their index field. Experiments show that the size of log file is greatly increased with the duplicate entries. Also there are entries which do not provide any information to log analysts. They solely increase the size of log file. Hence, the proposed approach uses the concept of event filtering to make the entries unique. It performs log mining on the existing log file. This uses the ItemSetCode algorithm to discover the frequent item sets. It uses a level-wise "candidate generate and test" method based on Apriori hypothesis. It collects the log, omits duplicate entries and orders them in a predefined order. As a result we obtain the frequent patterns in log file and then they can be omitted. This decreases the size of log file. Since the size is reduced, it normally reduces the uploading time and also the decryption time. This greatly enhances the performance of the system both in the case of time and space complexities. This implies that the system includes provisions for extracting log information as a whole and also it can access the log data with the unique values. This provision is provided with the Logging Client module.

Log File Size

The size of the log file is only depending on the number of entries within the log file. One of the major problems while considering log is that it usually includes a number of duplicate entries within it [11]. These entries only help to increase the size of the log. As the size of the log increases it takes more time for the logging client to prepare them as batches and to upload. Also it increases, the time being taken for decryption. In addition to duplicate entries, log files include standard informational entries also. They also affect the performance of the system like the other. Also, they do not provide any help to log analysts during their work. Hence, with our system we are removing such entries with the help of event filtering method. Thus, the proposed approach uses 2 methods for extracting log files:

- Non-optimized: In this method, the log data is being extracted as such. It includes all the entries within it.
- Optimized: In this method, event filtering is used so that, unique elements are being extracted. The resultant file doesn't contain any repeated elements.

The process of upload, retrieval, monitoring and deletion of log records are done with both optimized and non-optimized versions of log files. With the use of event filtering, it is possible to obtain two types of log files: One including entire log information and other including only the unique entries. The existence of these two kinds of files helps us to compare the space complexities in this context. The following figure shows a pictorial representation of the size comparison.

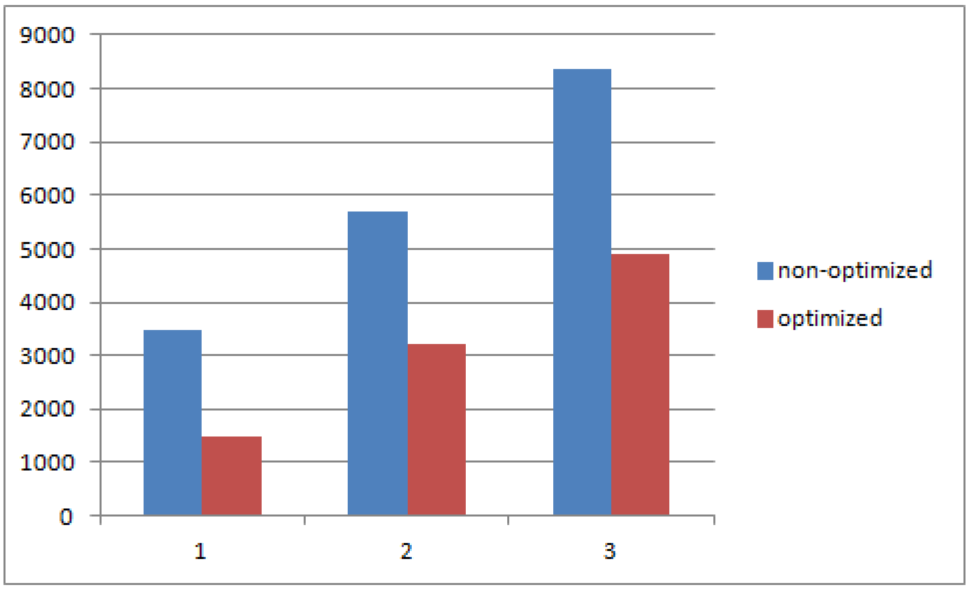

Figure 3 Comparing optimizing and non-optimizing log sizes

Here y axis represents the size of the log file in bytes. The system is tested in both optimized and non-optimized form. The log file sizes corresponding to non-optimized version are 3478, 5687 and 8345. For the same, when event filtering approach is used, we got the optimized version. This eliminated all duplicate entries. The corresponding values are 1489, 3211 and 4897. With this we can conclude that the method event filtering reduces the log size almost half. This is done not only by eliminating the duplicate entries, also by considering the entries within a constant time period as a single one. The result of event filtering is of greater impact as far as log management is concerned. As we discussed, Logging client prepares log batches as soon as it obtains the log file. If the log file is of huge size, then the time taken for preparing log batch is more. Otherwise it will be less. Also the size affects the upload time and decryption time of log data. This is because as the size increases there will be more number of batches and more entries. Hence the logging client takes more time to upload it. This is the case with decryption time also. Thus it is clear that the process of event filtering not only the size, but the time factor also is improved greatly.

6. Conclusion and Future Work

To decrease the log size, we introduced the concept of event filtering. It is a kind of log mining approach. The log file includes repeated entries as well as standard informational entries. These entries do not provide any advantage to the log analysts during their analysis. Hence it is desirable to omit such entries. The process of event filtering such entries and thereby it reduces the log size and makes it more useful. This helps the search to be more efficient Since the size of log file is decreased, there happens another important advantage in the area of time. With decreased log size. The time for batch preparation, uploading and decryption are greatly reduced. Thus with event filtering, the time and space complexities are greatly improved. In future, encryption of log records can be done in such a way that the logging server can execute some queries on the encrypted logs without breaching confidentiality or privacy. This will greatly reduce the communication overhead between a log monitor and the logging server needed to answer queries on logs. With the use of homomorphic encryption this can be achieved

References

[1] Indrajit Ray, Kirill Belyaev, Mikhail Strizhov, Dieudonne Mulamba, and Mariappan Rajaram. “Secure Logging As a Service—Delegating Log Management to the Cloud", IEEE Systems Journal, Vol. 7, No. 2, June 2013
[2] C. Lonvick, “The BSD Syslog Protocol”, Request for Comment RFC 3164,Internet Engineering Task Force, Network Work Group, Aug 2001.
[3] BalaBit, “Syslog-ng — Multiplatform Syslog Server and Logging Daemon", Multimedia Information Systems, pp. 56-65,2002.
[4] A. Shamir, “How to share a secret", Commun. ACM, vol. 22, no. 11, pp. 612–613, Nov. 1979.
[5] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. “Proactive secret sharing or: How to cope with perpetual leakage," in Proc. 15th Ann. Int. Cryptology Conf., Aug. 1995, pp. 339–352.
[6] Renáta Iváncsy, István Vajk, “Frequent Pattern Mining in Web Log Data", Proc. of ODBase‘04, Acta Polytechnica Hungarica Vol. 3, No. 1, 2006.
[7] K. Kent and M. Souppaya. “Guide to Computer Security Log Management", NIST Special Publication 800-92.

[8] D. Ma and G. Tsudik, “A new approach to secure logging", ACM Trans. Storage, vol. 5, no. 1, pp. 2:1–2:21, Mar. 2009.
[9] M. Bellare and B. S. Yee, “Forward integrity for secure audit logs", Dept. Comput. Sci., Univ. California, San Diego, Tech. Rep., Nov. 1997.
[10] Douglas C. Schmidt, “Scalable High-Performance Event Filtering for Dynamic Multi-point Applications", Department of Computer Science, Washington University, 1st International Workshop on High Performance Protocol Architectures, France, December 15-16, 1994.
[11] Bernard J. Jansen, “Search log analysis: What it is, what's been done, how to do it", Available online at www.sciencedirect.com, Library & Information Science Research 28 (2006) 407–432.

AUTHORS

Treesa Antony currently pursuing M.Tech in Computer Science and Engineering at Amal Jyothi College of Engineering, Kanjirappally, under Mahatma Gandhi University, Kottayam, Kerala, India. Also she completed her B.Tech in Computer Science and Engineering at Matha College of Technology, N. Paravur, under Mahatma Gandhi University, Kottayam, Kerala, India.