Distributed on Demand Logging using Secured Cloud Service

Jesheela A.P
Mtech Student, CSE Department
N.S.S College of Engineering
Palakkad

Sindhu S
HOD, CSE Department,
NSS College of Engineering
Palakkad

ABSTRACT
Log records are important part of an organization. Maintaining log records securely for a longer period of time is important for proper functioning of any organization. Since log files contain record of system events, the confidentiality and privacy of log data should be maintained and also integrity of log data and logging process should be ensured. The log data are stored in the server with in an organization for a fixed time and sent to the cloud .There will be a great chance of attack when log data are stored in plain text in the server of an organization. However, deploying a secure logging framework is one of the main difficulties that an organization faces in this new era. In this paper, we present an approach for secure logging by which log data can be sent to the cloud directly at run time.

Keywords
Logging, secure logging, cloud, rest, encryption

1. INTRODUCTION
Log records are important part of any organization. Logs are useful when performing auditing and forensic analysis, for identifying security incidents, establishing baselines, and identifying operational trends and long-term problems [1]. Since log data contain system crash results, user activities (e.g.: account details), logging is important and they become important targets of malicious attackers. An attacker tries to damage log files or log records, if the log files contain sensitive information then that will lead to confidentiality breaches. For example, in banking, the log information regarding transaction on account number of a particular user, can be helpful for an attacker to an unauthorized access to the system. Log information also leads to privacy/security violation of users in the system since log file contains record of all events in the system.

So logging should be provided in a secure manner and the log files should be protected from attackers. A Traditional logging protocol based on syslog [2] is not concerned with security features. However security extensions have been proposed in the paper [3], [4], and [5], [6]. But these do not protect the log records from end point attacks and only provides partial protection. Another important thing about log management is that the log service must be able to store data in an organized manner and provide a fast and useful retrieval facility and also log management requires substantial storage and processing capabilities too.Inorder to meet all these requirement an organization need to deploy a secure logging framework and it may needs huge amount .

In this paper, we propose a solution for storing and maintaining log records in a server operating in a cloud-based environment. Integrity, security and authentication can be maintained during log generation onwards. The major contributions of this paper are as follows. We propose architecture for secure logging and develop cryptographic protocols to address integrity and confidentiality issues with storing, maintaining, and querying log records at cloud provider. This prevents the cloud provider, or any other observer or attacker itself from manipulating the log data with other data. Finally, we develop a proof-of-concept prototype to validate our approach. So our approach provides a complete solution to the cloud based secure log management problem both practically and theoretically.

2. DESIRABE PROPERTIES OF SECURE LOGGING AS A SERVICE
The following are the desirable properties of securing logging service.

1) *Correctness:* Log data is useful only if it reflects true history of the system at the time of log generation.

2) *Tamper Resistance & verifiability:* Tampering is attacker's activity to alter/modify the data. A secure log must be tamper resistant in such a way that no one other than the creator of the log can introduce valid entries. Once those entries are created they cannot be manipulated without detection. The corresponding systems should be intelligent enough to make sure it's not tampered. The systems should check if some of the entries have been deleted.

3) *Confidentiality*: Log entries should not be stored in plain text format. Only the sender/receiver will be able to encrypt/decrypt the logged entries.

4) *Performance:* Since the application logging service is auxiliary functionality of any application, this should be fast enough. Otherwise it will affect the performance of the main application

3. RELATED WORKS
The one of the main property that is needed for secure logging service using cloud is correctness. That is the log data which is stored on the cloud should be correct, it should be exactly the same as the one that was generated. The next desirable properties are the tamper resistance – log data cannot change or modify once they are generated without any detection. The main advantage of secure log is that no one can including the attacker can change the log data and any attempt to alter log data [7] would be failed .The other important properties are confidentiality and security. Log records should not be searchable to gather sensitive information other than system administrator. Moreover log records should not be traceable or linkable to their sources during transmission and in storage.

A number of approaches have been proposed for secure logging. Syslog-ng [4] application supports reliable and encrypted transmission of log messages using TCP and TLS/SSL. It supports IPv6, reliable transfer log messages using TCP, and filtering the content of logs using regular

expressions. In this ,in order to ensure integrity and confidentiality log record are encrypted using SSL during transmission. But it does not protect log data form modification at end points. An enhancement to Syslog-ng is syslog-sign[9].It provide authentication, message integrity, tamper resistance, message sequencing, and detection of missing messages using two additional messages—signature blocks and certificate blocks. But it does not provide confidentiality or privacy during the transmission of data or at the end points.

In Syslog-pseudo [10] log records are first processed by a pseudonymizer before being collected. The pseudonymizer filters out identifying features from specific fields in the log record and substitutes them with carefully crafted pseudonyms. So the log records that are stored are not the same as the ones that are generated. So this paper doesn't ensure correctness. Another problem with this paper is that while the protocol anonymizes each log record individually it does not protect log records from attacks that try to correlate a number of anonymized records. On the other hand, our objective is precisely this. Furthermore, privacy breaches that can occur from scenarios such as the user erroneously typing the userid in a password field (as discussed earlier) or identifying information available in fields that are not anonymized, are also not addressed in this paper. In paper [11], The anonymous log file anonymizer performs a similar anonymization of identifying information by substituting with default values . But, the problem with this paper is that the original values cannot be restored. Syslog-pseudo, anonymous log file anonymizer couldn't protect log records from confidentiality and integrity violations and other end-point attacks.

Paper [6] aims to implement reliable delivery of syslog messages and is built on top of the blocks extensible exchange protocol (BEEP [12]) which runs over TCP to provide the required reliable delivery service. This protocol allows device authentication and incorporates mechanisms to protect the integrity of log messages and protect against replay attacks of log data; however it does not prevent against confidentiality or privacy breaches at the end-points or during transit.

The method of forward-integrity of log records was proposed by Bellare and Yee [13] to protect log data from post compromise insertion, deletion, modification, and reordering. This is established by a secret key that becomes the starting point of a hash-chain. In this, the hash-chain is generated by a cryptographically strong one-way function and the key is changed for every log record. Schneier and Kelsey [14] also proposed a logging scheme that supports forward integrity. It is based on forward-secure message authentication codes and one-way hash chains similar to that suggested by the Bellare-Yee protocol. But, the major problem of both schemes is that both require online trusted servers to maintain the secret key and verification of log records. When the trusted server is attacked or compromised, then the security of the log record is broken. Holt [15] incorporates public verifiability of log records. This scheme being a public-key based scheme, the overhead is significantly more. But these three schemes don't consider the privacy concerns of storing and retrieving log records. All these three scheme suffer from truncation attacks where an attacker deletes a contiguous subset of log records from the very end. In paper [7] Ma and Tsudik address this problem and they use the notion of forward-secure sequential aggregate authentication in which individual signatures are folded into one single aggregated signature and all other signatures are deleted. Without knowing all previous signatures an attacker cannot recreate this signature. Ma and Tsudik's scheme is very expensive to verify only a single log record.

In paper [16] secure logging service implemented through batch updates .This scheme include log generator, logging client, log monitor and logging cloud. Log generator generates log data in plain text format files. Log clients take this file and encrypt and sent to the cloud in batches. The disadvantage of this approach is that the log data is stored as plain text for a short while.

4. PROPOSED SYSTEM ARCHITECTURE

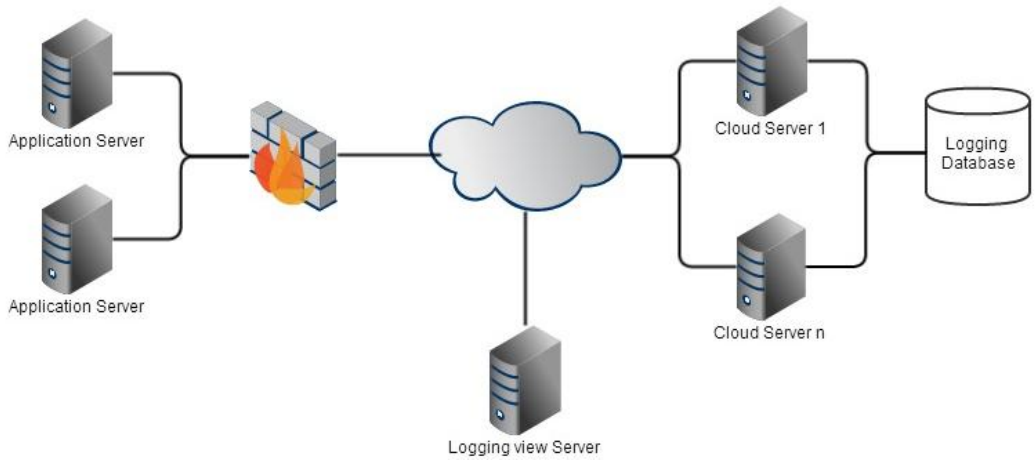


Fig 1: Proposed System Architecture

The proposed system architecture consists of on premise servers and cloud infrastructure. The application servers are at on premise and the logging service and database at cloud infrastructure.

- Application server: These are the logging clients which create the logs on execution of the process. This will be on premise servers. In memory log data will be encrypted and invoke the cloud based logging service to store the log entries.
- Cloud server: These servers will be responsible for providing the provision for storing /retrieving the log data in/from the database. We are using cloud

facilities for both for web servers and database.

- Log viewer server: This will be brand new application for retrieving the encrypted hashed logged data from the cloud hosted service. This will be responsible for verifying the log content against tampering and decrypt the logged entries.

4.1.1 Add Log Architecture

4.1 Application architecture

The application architecture is divided into two.Viz, Add log architecture (dealing with logging client and cloud logging add service) and retrieve log architecture (dealing with log retrieve viewer and cloud log retrieve service)

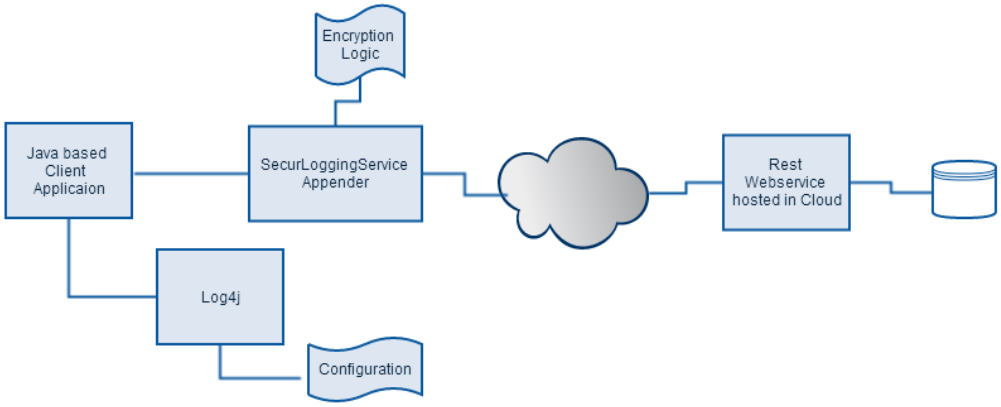


Fig 2: Application architecture

4.1.1.1 Modules

- Java based console Application: This Java application will be proof of concept console application which performs logging operation.
- SecureLoggingServiceAppender for log4j: Log4j [24] supports add-ons to its framework as appender. This will be a custom appender which performs secure logging using a service.

4.1.2 Data flow diagram

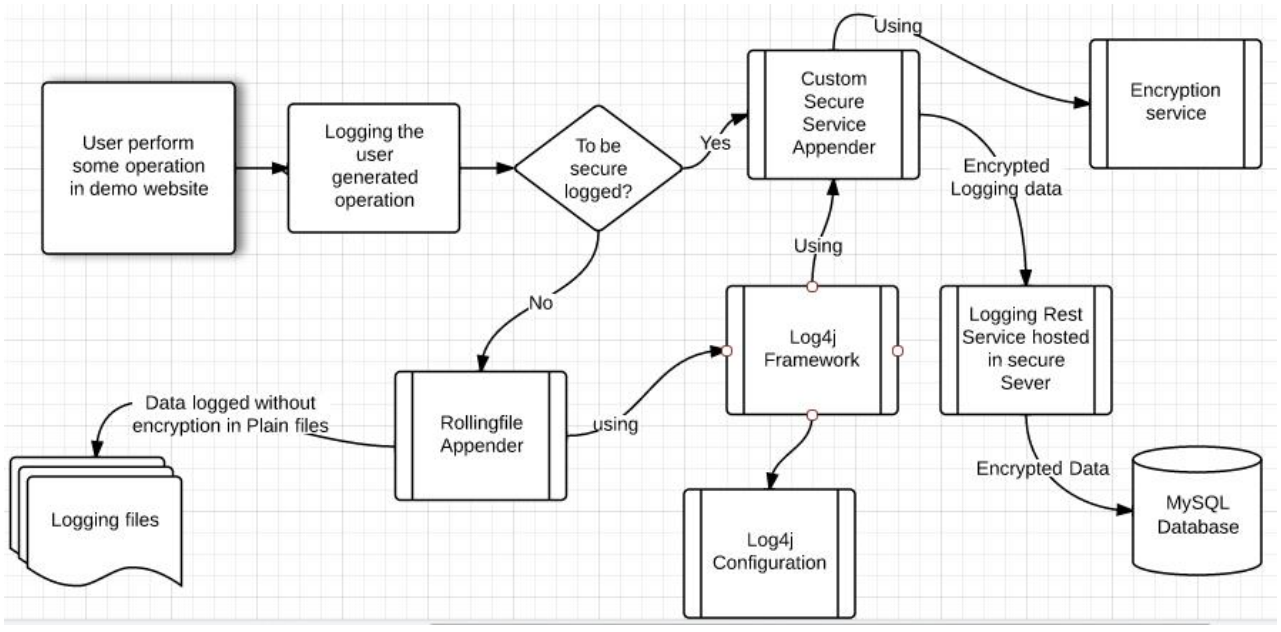


Fig3: Data flow diagram

The logging client is the client application that trying to log the entries. There is no log generator here. Client application will decide the log should be secure or not. If the log needs to be secured, then the new program that has written on top of Log4J framework will be kicked off. The custom secure service appender is the new appender what we are going to create. This will be using the encryption logic (AES implementation in JAVA). In the log4j configuration , we can register this new appender. After encrypting the content it will invoke the rest based web service which has been configured.

The REST based service hosted in the cloud will provide the entry point for logging the data in the cloud based database. Before logging it will perform hashing to make the entries tamper proof.

4.2 Retrieve log architecture

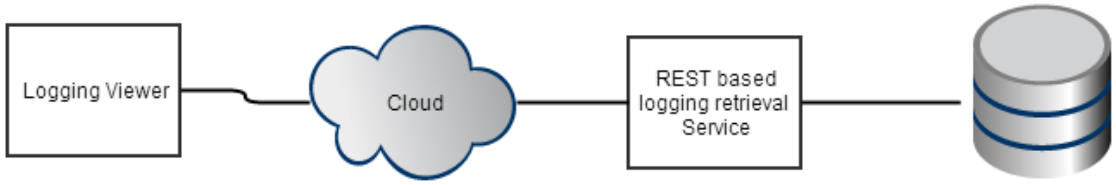


Fig 4: Architecture of the log data retrieval

The log viewer is a console based application which invokes the REST based http cloud service to retrieve the encrypted hashed entries. This application will perform the decryption and hashing for verification once it got the logs from the cloud service.

In the development environment, we will develop 3 applications

1) Logging client application: This will be a proof of concept console application to showcase the secure logging. We will incorporate log4j framework for logging and our new secureloggingservice appender to that.
2) Logging services REST based: we will create a brand new REST based web service to log the content in database. This will be using only HTTP POST. We also have one service which will get the content from logging database. This will be http GET. We will be using Tomcat, spring MVC, Java and MYSQL as software/tools.
3) Log viewer Application: This is a console based application which invokes cloud based retrieval REST service. After getting the content it will perform the hashing for tamper proof verification and decryption to show the log entries.

4.3 Tools

- Apache log4j [22] [23] is a Java-based logging utility. It was originally written by Ceki Gülcü and is now a project of the Apache Software Foundation. log4j is one of several Java logging frameworks. There are three ways to configure log4j: with a properties file, with an XML file and through Java code. Within either you can define three main components: Loggers, Appender and Layouts. Configuring logging via a file has the advantage of turning logging on or off without modifying the application that uses log4j. The application can be allowed to run with logging off until there's a problem, for example, and then logging can be turned back on simply by modifying the configuration file.
- Spring MVC : [19] [20] The Spring Web model-view-controller (MVC) framework is designed around a Dispatcher Servlet that dispatches requests to handlers, with configurable handler mappings, view resolution, locale, time zone and theme resolution as well as support for uploading files.
- Tomcat Server: Apache Tomcat is an open source software implementation of the Java Servlet and Java Server Pages technologies. The Java Servlet and Java Server Pages specifications are developed under the Java Community Process.
- Jelastic Cloud: Jelastic is a Platform-as-Infrastructure (PAI) cloud computing service that provides networks, servers, and storage solutions to software development clients, enterprise businesses, OEMs and Web hosting providers.
- Apache HTTP client libraries: These libraries are used to invoke cloud based REST services.
- MySQL DB: We are using MySQL database to store the log entries and this will be in the cloud behind the service.

5. ALGORITHM AND FLOW

5.1 Logging client

Logging client is s a proof of concept implementation to mimic the logging application. Ideally this module can be implemented any enterprise application that requires secure logging. When we consider the logging libraries, log4j (java) and log4net (.net) are the main libraries that are used for logging. The advantages over other libraries are these libraries are easier to configure using properties file. With use of properties file, this logging source and behavior of the logging can be changed. The other aspect is that it is highly customizable by using the interceptors.

We are using the interceptor extension provided by Log4J to log the loggings to a rest service. In the properties file we will provide the http POST url to log the service. We have written a custom appender that will be invoked for all log requests through log4j.

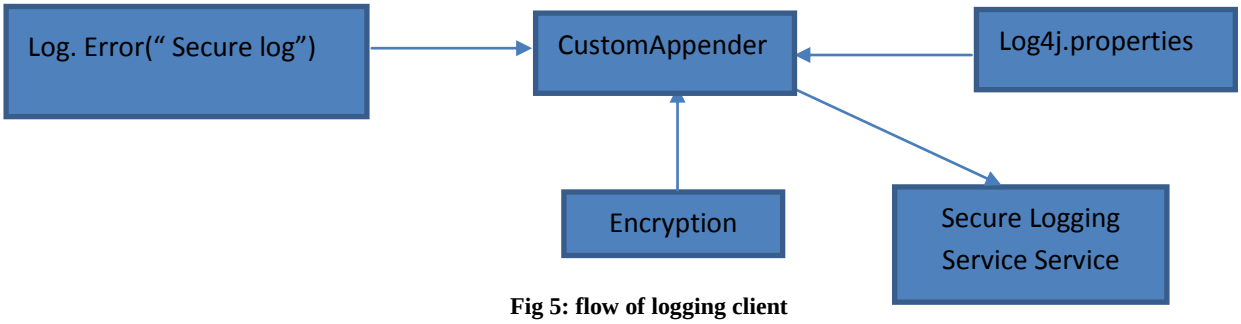


Fig 5: flow of logging client

The content that is getting the appender will be encrypted using "AES" symmetric encryption algorithm. This encrypted content will be send to the secure log service using HTTP POST technique.

L1 → Encrypted SHA256 algorithm [Private Key] → E (L1)

L2 → Encrypted SHA256 algorithm [Private Key] → E (L2)

L3 → Encrypted SHA256 algorithm [Private Key] → E (L3)

……

Ln → Encrypted SHA256 algorithm [Private Key] → E (Ln)

This encrypted log will be sending across to rest based , secure logging service.

Pseudo Code

1. Get the log message using log4j interceptor
2. Convert the string message to bytes.
3. Pass the bytes to AES encryption algorithm along with the private key.
4. Get the encrypted message from the algorithm.
5. Get the cloud service endpoint (add secure logs) from the properties URL.
6. Create new HTTP client using apache library.
7. Create a post method and set the encrypted data into that.
8. Invoke the post method.
9. Make sure we are getting 200 OK as HTTP response for HTTP post call.

5.2 Secure logging service

5.2.1Secure Logging Add service

The secure logging service is spring MVC rest [18] based application. We have implemented it as POST implementation. This will accept encrypted logging content. This log service will store the content in database. The content storing in the database is hashed encrypted content.

In database we have created new log table having auto increment integer as primary key. The other columns in the table are Message to store the encrypted hashed message and the current system date time.

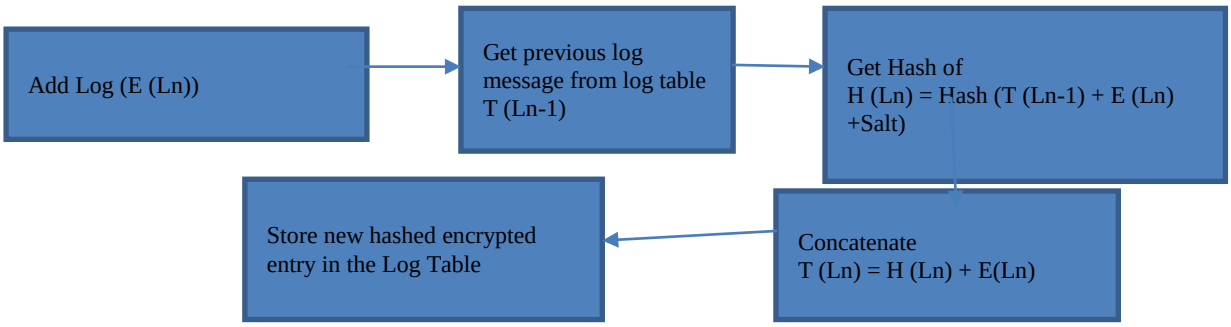


Fig 6: Flow of secure logging service

Pseudo Code

1) The logging client will invoke rest based http POST service by providing the encrypted Entry. E(Ln)
2) As soon as we get the request , it will get the last entry from the Log table -- T(Ln-1)
3) Using SHA 512 hash algorithm compute hash by concatenating T(Ln-1) , E(Ln) and Salt
4) Salt is act like one more level of security this will ensure even you are getting the content you will not be able to produce the same digest (output from hash)
5) If the given log is the very first log then H(L1) = Hash(E(Ln)+Salt)
6) The output of hash is UTF8 encoded 32 byte (256 bit) digest.
7) Convert the digest to hexa decimal encoding (64 byte) we termed it as H(Ln)
8) After that we will concatenate H (Ln) with E (Ln). We termed it as T(Ln)
9) The T(Ln) will be inserted into the Log table with system date time and Identity key(auto increment)

5.3 Secure logging retrieve service

Logging retrieve service is part of the secure logging service. Its duty is to provide the encrypted hashed logs to the clients which have been invoked for the same. This has been implemented as GET http Rest service [17] .

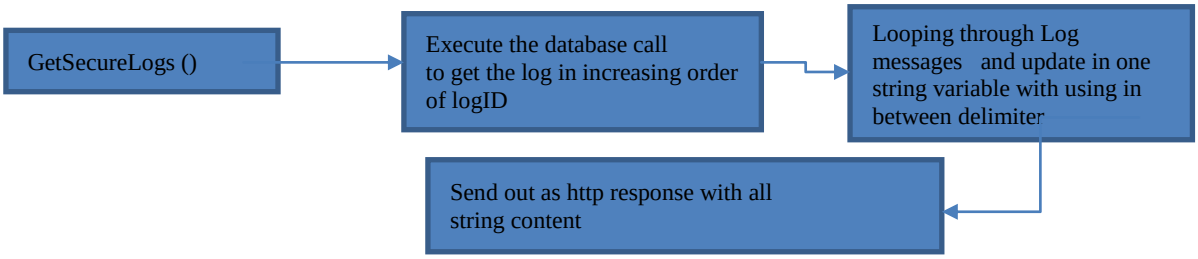


Fig 7: Flow of log retrieval service

Pseudo Code

1. Log View Client will invoke this service to get the secure logs .
2. This service will fire a database call to get the logs from the Log table.
3. After getting the resultset in loop through all messages add to the output string.
4. Use "#Secure Message#" as a delimiter.
5. So the message will look like T(L1) + "#Secure Message#" + T(L2) +……+ T(Ln)
6. Put this concatenated jumbo string to http response.
7. The Log Viewer client will get this response.

5.4 Log viewer Client

The purpose of the Log Viewer client is to get the encrypted hashed logged messages to readable format for its authenticated users. It has to make sure the logs are not tampered before decrypting the messages. If one of the messages are tampered then need to tell the user that "it is broken"

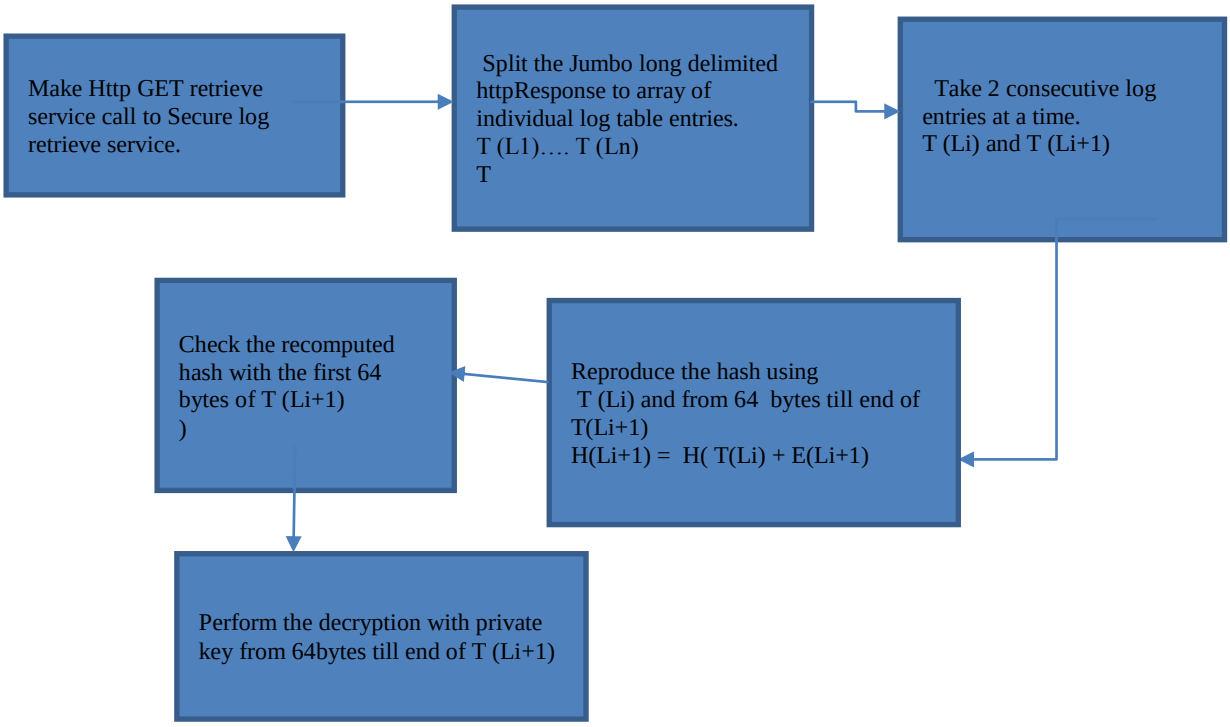


Fig 8: Flow of log viewer

Pseudo Code

1. This log view client will make a HTTP GET request to the Log retrieval service to get the encrypted hashed log entries.
2. Split by using the delimiter "#seclogMessage" to string of array.
3. This array will be log entries T (L1)…… T (Ln).
4. T(Li+1) = H(T(Li)+ E(Li+1)+ salt) + E(Li+1) This can be also defined as Tx = [64 bytes of hash digest] + [EncryptedLogEntry]
5. Compute the hash using H (Li+1x) =T(Li)+ E(Li+1)+ salt
6. Get the first 64 bytes of T(Li+1) , H (Li+1)
7. If H (Li+1x) equals H(Li+1) then no one has tampered it . Logs are good.
8. If those are not equal then logs are broken.
9. For good log, take rip off first 64 bytes of log entry then decrypt it using the private key.
10. Show the decrypted message to the client.

6. IMPLEMENTATION AND PERFORMANCE

As we mentioned earlier our implementation is using java related technologies. For cloud we have used jelastic cloud (which is fall under PAS model of the cloud) [22]. We have used Eclipse Kepler as integrated development environment for developing all java applications (console/ services) for the project. All together our code base consists of ~ 300 lines of code. We were using MySQL as our database.

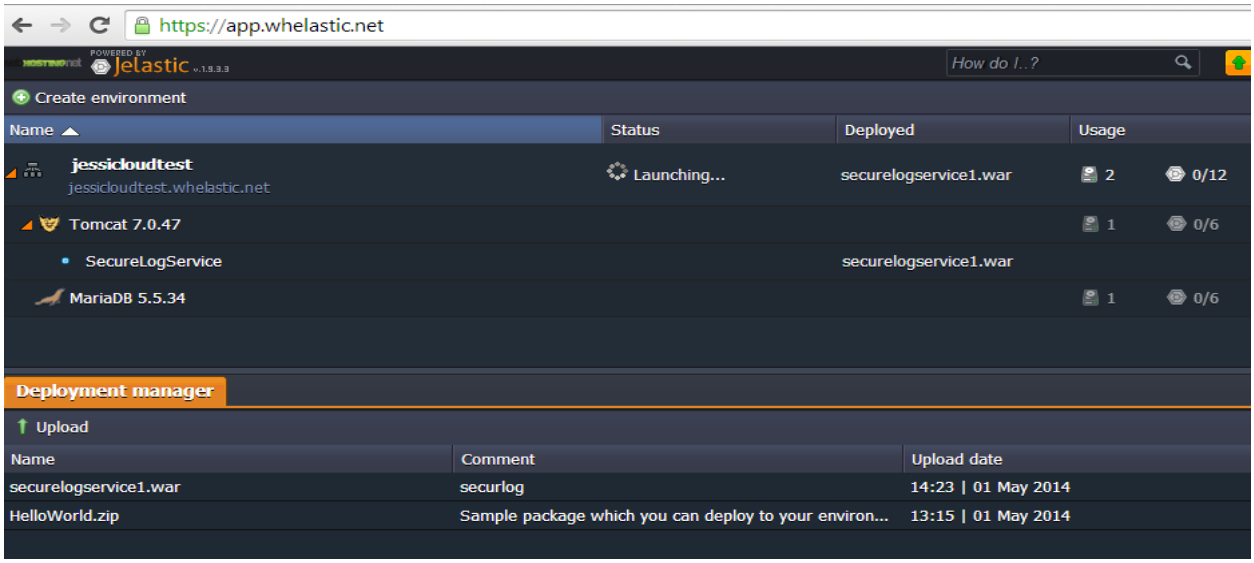


Fig 9: jelastic UI for secure log service deployed in tomcat

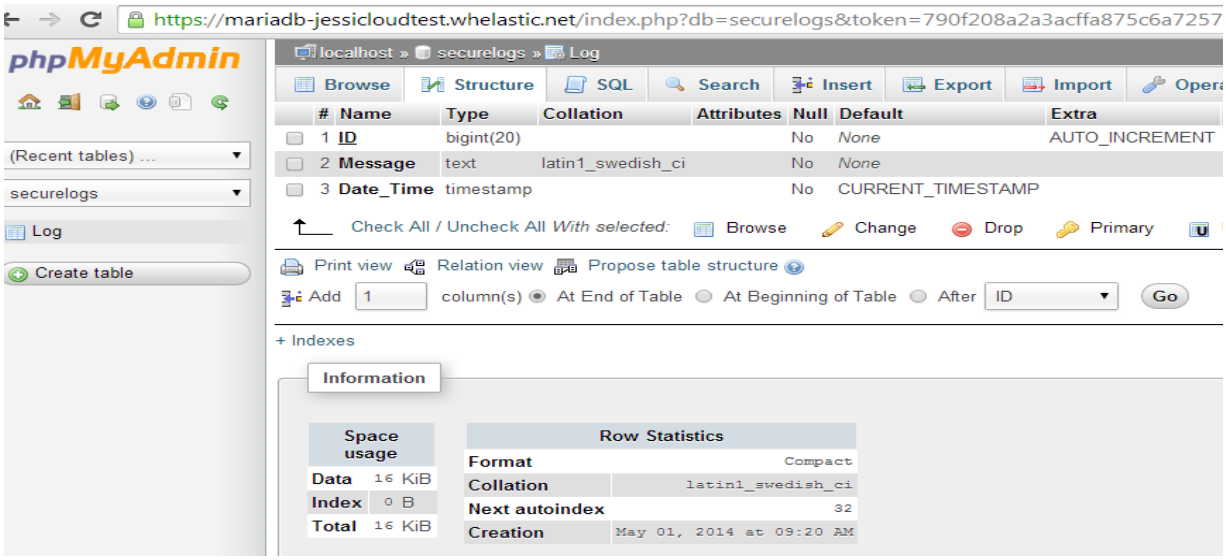
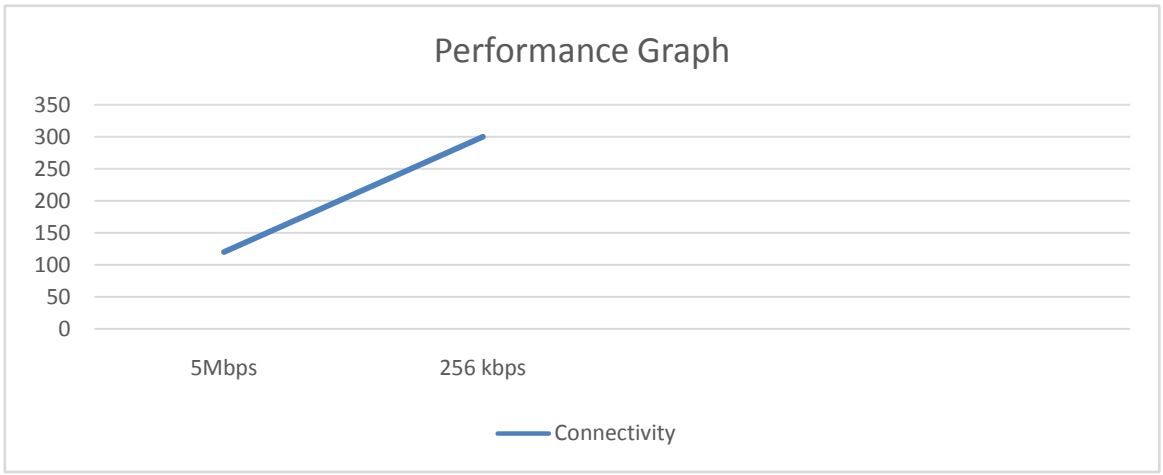


Fig 10: Structure of table in jelastic cloud

6.1 Performance

Since logging service will be invoked for each and every logging we need to have good connectivity. On 5mbps internet service we were able to see <130 msec response time. On 256 kbps connectivity it has been seen around 300 msec to complete one transaction of secure logging. In order to minimize the latency we can think of the cloud service hosted near to the geography of the main logged application.



Advantages over Base paper implementation

The main advantages over the base paper implementation are

1) In base paper, logging has been implemented in batch mode. The data is stored plain and then later using secure batch service pushing to the cloud. Whereas this implementation is log from the source.

2) In base paper they have made complex implementation of secure cloud logging. Whereas in proposed implementation it's very easy to configure logging with existing projects.

3) It's not easy to switch between the logging needs from application stand point in the base paper implementation. Whereas it's super easy to change the logging needs. It can be done just by flipping the properties file.

4) At any point of time data will not be in plain text format

7. CONCLUSION AND FUTUREWORK

Logging plays a very important role in the proper operation of an organization's information processing system. However, maintaining logs securely over long periods of time is difficult and expensive in terms of the resources needed. The emerging paradigm of cloud computing promises a more economical alternative. The existing system proposed a complete system to securely outsource log records to a cloud provider. The existing system identified problems in the current operating system based logging services practical difficulties in some of the existing secure logging techniques. Since the existing system uses batch process and at first the individual applications log the data are saved in plain text format in files. So someone who can access the system can see the log data and it does not provide transport level secuirity.The proposed system overcome the disadvantages of the existing system by transferring the encrypted log data at run time.

In the proposed system, logging client is invoking the service (cloud based secured service) just by sending the encrypted log data. Since there could be 'n' number of logging clients are trying to invoke the service, there could be synchronization issues (slow response). In order to avoid that unique id info can be passed along with the service .This will ensure to identify the correct logging client details at the service side and use it one of the attribute to log the data. Currently the logging retrieval processes are not optimized. The suggestion is to implement date and time based logging retrieval processes. Retrieval based on logging client server specific retrieval can be think of further enhancements in this area.

8. REFERENCES

[1] K. Kent and M. Souppaya. (1992). *Guide to Computer Security Log Management, NIST Special Publication 800-92*[Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf

[2] C. Lonvick, *The BSD Syslog Protocol*, Request for Comment RFC 3164, Internet Engineering Task Force, Network Working Group, Aug. 2001.

[3] M. Bellare and B. S. Yee, "Forward integrity for secure audit logs," Dept. Comput. Sci., niv. California, San Diego, Tech. Rep., Nov. 1997.

[4] BalaBit IT Security (2011, Sep.). *Syslog-ng—Multiplatform Syslog Server and Logging Daemon* [Online]. Available: http://www.balabit. com/network-security/syslog-ng

[5] J. Kelsey, J. Callas, and A. Clemm, *Signed Syslog Messages*, Request for Comment RFC 5848, Internet Engineering Task Force, Network Working Group, May 2010.

[6] D. New and M. Rose, *Reliable Delivery for Syslog*, Request for Comment RFC 3195, Internet Engineering Task Force, Network Working Group, Nov. 2001.

[7] 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.

[8] Shams Zawoad, Amit Kumar Dutta& Ragib Hasan "SecLaaS: Secure Logging-as-a-Service for Cloud Forensics", ACM Trans. Inform. Syst. Security, vol. 2, no. 2, pp. 159–176, May 1999.

[9] J. Kelsey, J. Callas, and A. Clemm, *Signed Syslog Messages*, Request for Comment RFC 5848, Internet Engineering Task Force, Network Working Group, May 2010.

[10] U. Flegel, "Pseudonymizing unix log file," in *Proc. Int. Conf. Infrastruture Security*, LNCS 2437. Oct. 2002, pp. 162–179.

[11] C. Eckert and A. Pircher, "Internet anonymity: Problems and solutions,"in *Proc. 16th IFIP TC-11 Int. Conf. Inform. Security*, 2001, pp. 35–50 .

[12] M. Rose, *The Blocks Extensible Exchange Protocol Core*, Request for Comment RFC 3080, Internet Engineering Task Force, Network Working Group, Mar. 2001.

[13] M. Bellare and B. S. Yee, "Forward integrity for secure audit logs," Dept.Comput. Sci., Univ. California, San Diego, Tech. Rep., Nov. 1997.

[14] B. Schneier and J. Kelsey, "Security audit logs to support computer forensics," *ACM Trans. Inform. Syst. Security*, vol. 2, no. 2, pp. 159–176, May 1999.

[15] J. E. Holt, "Logcrypt : Forward security and public verification for secure audit logs," in *Proc. 4th Australasian Inform. Security Workshop*, 2006,pp. 203–211.

[16] 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

[17] http://en.wikipedia.org/wiki/Web_service

[18] http://spf13.com/post/soap-vs-rest

[19] http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[20] http://viralpatel.net/blogs/tutorial-spring-3-mvc-introduction-spring-mvc-framework/

[21] http://en.wikipedia.org/wiki/Jelastic

[22] http://www.java-logging.com/

[23] Log4J: http://logging.apache.org/log4j/1.2/

[24] http://en.wikipedia.org/wiki/Log4j