Universidad Zaragoza
1542

PROYECTO FIN DE CARRERA
de
INGENIERÍA INFORMÁTICA

Registro de Logs firmados de SSII de alto rendimiento con valor jurídico según la ley 59/2003

(Log4Legal v.1)

Autor: Jose Antonio Ortiz Bascuas
Director: Elvira Mayordomo Cámara
Departamento: Informática e Ingeniería de Sistemas
Fecha: 27 de Febrero de 2015

Registro de Logs firmados de SSII de alto rendimiento con valor jurídico según la ley 59/2003

RESUMEN

Descripción de la situación actual y requisitos de partida para la creación de una herramienta de trazabilidad, en tiempo de ejecución, de aplicaciones software. En este punto, el lector debe conocer la utilidad, necesidad y usabilidad de este tipo de herramientas software.

Principios forenses, de ámbito lo mas general posible para su aplicación en otros países. Introducir la necesidad de una herramienta que produzca trazas en sistemas de aplicaciones software que sean admisibles en un tribunal de justicia.

Estudio de la legalidad e interpretación de la ley orientada al tipo de herramientas en cuestión. Introducir el requisito "legalidad" en la discusión y que forme parte de las decisiones sobre la funcionalidad y el listado final de requisitos de usuario la "nueva" herramienta.

Entendidas las necesidades y obligaciones de la entidad para con la legalidad de las trazas que generan sus sistemas de información, deben conocerse las alternativas existentes en el mercado, hasta donde llegan y si alguna de estas necesidades u obligaciones quedan desiertas. Una vez justificada la implementación de una nueva herramienta (*log4legal)*, se usará el análisis realizado de las aplicaciones existentes para rellenar el hueco que han dejado y se aplicarán sus éxitos para realizar el diseño.

Enumeración de los requisitos técnicos para *log4legal*: funcionales, rendimiento, explotación, usabilidad, mantenimiento, arquitectura y seguridad.

Análisis Técnico y Casos de Uso de *log4legal*.

Criterios de Aceptación para la herramienta *log4legal*.

Proyecto git para los desarrollos y ejemplos incluidos en este PFC:
https://github.com/jaortizb/Log4Legal.git

ÍNDICE

EDICIONES Y REVISIONES

Edición	Fecha	Observaciones (aquí deberán identificarse de forma sucinta los cambios que conlleva la nueva edición)
1.00	27.11.15	Versión inicial

TABLA DE REVISIONES Y APROBACIONES

ELABORACIÓN	REVISIÓN	APROBACIÓN
José Antonio Ortiz Bascuas		

INTRODUCCIÓN

El registro logs está a menudo relegada a una actividad de un segundo plano, mientras se tienen en cuenta aspectos relativos a la arquitectura, diseño y desarrollo de aplicaciones. Finalmente, cuando la aplicación está lista, se despliega. Y entonces, ¿qué sucede?

Su aplicación ahora está fuera de su entorno de desarrollo, no está en su banco de trabajo IDE favorito y mucho menos dispone de un depurador a su servicio. Ahora que te das cuenta de la importancia de lo que implica el registro de logs. Y a medida que manejas de un enorme conjunto de registros que tratan de depurar cualquier problema con la actividad normal de tu aplicación, te das cuenta de que no es una tarea trivial, es aburrido y tedioso. Uno se pregunta, aún con toda experiencia y conocimientos, si esto lo que estabas destinado a hacer, ¡encontrar una aguja en un pajar de troncos!.

Todas las empresas que realizan la explotación de SSII, de prácticamente cualquier tamaño, necesitan que sus aplicaciones dispongan de una "bitácora" donde se registran los eventos del sistema. Este registro permite la explotabilidad en cuanto permite registrar información sobre quién, qué, cuándo, dónde y por qué un evento ocurre para un dispositivo o aplicación en particular. Esto hace posible, por ejemplo, la supervisión del sistema y la localización de patrones de error y resolución de incidencias.

Para facilitar la comprensión de los conceptos introducidos podemos pensar como se aplicarían para un sistema SSII determinado y conocido. Pondremos un servidor Web como ejemplo, que nos servirá en la mayoría de los casos.

La palabra log, o traza, se relaciona también con el término evidencia digital. Un tipo de evidencia física construida de campos magnéticos y pulsos electrónicos que pueden ser recolectados y analizados con herramientas y técnicas especiales, lo que implica la lectura del log que deja al descubierto la actividad registrada en el mismo.

El registro y la validez de esta "evidencia digital" serán el objetivo de este proyecto. Y enfocando un poco mas el ámbito de aplicación, el objetivo es llevar este proyecto (o al menos hacer compatible con) al campo de los SSII de Alto Rendimiento/Disponibilidad y las restricciones/limitaciones inherentes al mismo.

En el panorama actual las aplicaciones en estudio generan trazas muy útiles que ayudan a resolver todos los problemas para los que han sido diseñado. Pero fallan a la hora de darles validez, sobre todo legal, por lo falsificables que resultan. Como primeros puntos de esta memoria se establecen los conceptos y conocimientos que nos ayudarán a tomar un criterio para el análisis de su resolución.

Tampoco debe olvidarse que en el material expuesto así como las decisiones de implementación a lo largo de todo el proyecto, debe hacer referencia a sistemas en el ámbito del Alto Rendimiento. Esto es así, porque es en este ámbito donde aparece, de forma obligatoria el rol del Responsable de Seguridad, ademas de que son estos los sistemas que usan de forma masiva el registro de eventos.

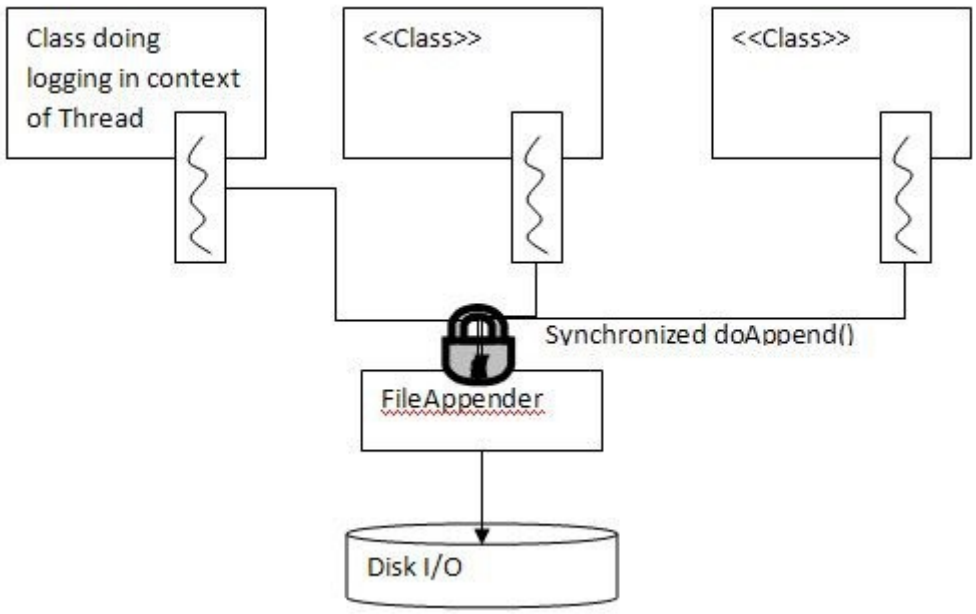
Básicamente, este proyecto proporcionará una herramienta útil y que, a día de hoy no existe, al Responsable de Seguridad. Es a él a quien van a echar la culpa del resultado cualquier ataque/violación al SSII de su competencia, si no tiene definido una política de seguridad, o si esta es insuficiente, y será su labor/valor en la explotabilidad la que quede en entredicho.

En términos generales deben de tenerse en cuenta, en mayor o menor medida, los siguientes factores, para confeccionar un sistema de registro de logs:

Monitoreo: La aplicación mas frecuente de un sistema de registro es el Seguimiento. Ayuda a que el desarrollador de aplicaciones pueda llegar a la causa raíz de los errores de aplicación. Pero a menudo se hace para ser la única aplicación de Monitoreo, posiblemente porque sea la opción menos invasiva a la aplicación. Sin embargo, el registro excesivo de logs puede conseguir el efecto contrario al propósito mismo del registro.

Rendimiento: Se pregunta a cualquier especialista en rendimiento o arquitecto y en casi el 90% de las solicitudes, el registro excesivo tiene un efecto muy negativo en el rendimiento. El registro es un O / I intensa actividad y puede causar graves perjuicios al rendimiento de la aplicación, sobre todo si al iniciar la sesión se hace de la manera tradicional, es decir, de forma sincrónica por escrito a un FileAppender.

No sólo eso, el método doAppend () que se llama internamente para registrar a un Log4J Appender, se sincroniza para proporcionar la capacidad “Thread safe” a la herramienta de registro. Lo que esto significa es que los hilos de aplicación no sólo están haciendo un uso pesado de llamadas I/O sino que también unos hilos bloquean a otros mientras se escribe en el registro.



Seguridad (auditoría y otra información confidencial): Los registros de auditoría son una clase especial de registros que permitan a la auditoría de seguridad de la aplicación, y se utilizan para

realizar un seguimiento de las acciones de los usuarios. Este es un ejemplo de cómo el registro ayuda a las prácticas de seguridad.

En el otro extremo, tenemos que, si los registros contienen información confidencial, como contraseñas de cuentas de usuario, estos pueden exponer a una vulnerabilidad al sistema.

Además, deberemos garantizar y hacernos responsables, si existen garantías, de la validez de los registros. Esto que parece evidente tiene consecuencias dramáticas en la información que deberemos guardar en cada registro y nos obliga a revisar las bondades de tener una política de almacenamiento seguro y extracción, como si fueran, evidencias electrónicas del funcionamiento de nuestro sistema. Y esto a su vez nos obliga a revisar los principios para la actuación forense y la ley vigente a propósito de la información de registro.

Escalabilidad y alta disponibilidad. Al mejorar el rendimiento de una aplicación con técnicas de "alto rendimiento" y un registro de calidad, debe tenerse en cuenta la mejora de la escalabilidad y la disponibilidad de la aplicación con menos hardware, y usar toda esta potencia para la funcionalidad "no administrativa" de la aplicación. Así pues, si su aplicación es capaz de hacer una extracción más pesada con los recursos existentes, tendrá menores motivos para fallar.

0.1 Ejemplos de uso de logs

A modo de demostración, se adjuntan sendos fuentes relativos a los diferentes paradigmas de registros de logs usados de forma más común.

En apartados posteriores se explicará con mas detalles las características de estos sistema de registro, pero en el código que se muestra a continuación se refleja la facilidad de uso de log4j:

```
import org.apache.log4j.Logger;
import java.io.*;
import java.sql.SQLException;
import java.util.*;

public class log4jExample{
      static Logger log = Logger.getLogger(log4jExample.class.getName());
      public static void main(String[] args) throws IOException,SQLException{
            long start_time = System.currentTimeMillis();

            for(int i=0;i<1000000;i++)
                  log.debug("Hello world!");

            long end_time = System.currentTimeMillis();
            System.out.println(end_time - start_time);
      }
}
```

En este ejemplo se mide el registro de un millon de veces de un mismo registro de log "Hello world ... ", y con una configuración como la siguiente:

```
# Define the root logger with appender file
log = .
log4j.rootLogger = DEBUG, FILE
```

```
# Define the file appender
log4j.appender.FILE=org.apache.log4j.FileAppender
log4j.appender.FILE.File=${log}/log.out

# Define the layout for file appender
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE.layout.ConversionPattern=%d{yyyy-MM-dd}-%t-%x-%-5p-%-10c:%m%n
```

produciría una salida "2015-03-19-main—DEBUG-log4jExample: Hello world!" un millon de veces, cada una de ellas volcada sobre el fichero *log.out.*

Tampoco podemos dejar de mencionar el uso de syslog, del que a continuación se adjunta el mismo ejemplo "hello world!":

```
#include <stdio.h>
#include <unistd.h>
#include <syslog.h>

int main(void) {
      openlog("slog", LOG_PID|LOG_CONS, LOG_USER);
      syslog(LOG_INFO, "Hello world!");
      closelog();

      return 0;
}
```

En ambos casos se ve claramente que son fáciles de usar como herramienta de seguimiento pero la información generada debe cumplir una serie de requisitos que aseguren su calidad, desde el punto de vista de seguridad y rendimiento, y que todavía están por valorar.

OBJETO

El objeto de este documento es definir los requisitos de Funcionales y de Sistemas para cubrir la necesidad de un producto de las características descritas.

Este proyecto estará comprendido dentro de una política de seguridad para la trazabilidad de aplicaciones de sistemas de alto rendimiento.

Esta memoria analizará las necesidades, los distintos problemas, recursos y alternativas, que una empresa a cargo de aplicaciones, en un entorno de alto rendimiento, debería de tomar en cuenta para implementar las herramientas que correspondan para favorecer una política de seguridad viable que permitan el registro de las evidencias digitales que las diferentes aplicaciones emitan.

Este proyecto también deberá exponer las necesidades legales que son necesarias para que las evidencias descritas puedan incluirse en una investigación forense sin posibilidad de rechazo.

Con las conclusiones de los análisis descritos se realizará una herramienta con la funcionalidad básica y flexible, con licencia por determinar, de código abierto para que los SSII que tengan a bien implantar esta herramienta tengan la posibilidad de adaptarla a sus peculiaridades.

UNIDADES AFECTADAS

El documento va dirigido a:

- Desarrollo de Sistemas
- Seguridad Informática.
- Arquitectura.
- Producción/Explotación de Sistemas.

1 PROCESO DE CLIENTE/INTERNO ORIGEN

Este apartado tiene por objeto la definición de conceptos que generan la preocupación y la necesidad que justifica una herramienta de la que se está planteando su construcción.

La solución que busca este proyecto está acotado sobre un sistema de alto rendimiento y disponibilidad. Los sistemas de logs disponibles son muy genéricos y amplios en su configurabilidad y no tienen en cuenta estas premisas de eficiencia ni ningún otro aspecto de tipo legal.

Para no perder este referente, y como punto de partida, pasa a describirse la legalidad general que todo analista forense debe conocer y llevar a la práctica a la hora generar sus informes y operaciones. De esta descripción nace la preocupación por los sistemas de logs “tradicionales” y su adecuación a estos principios.

1.1 Sistemas de gestión de las evidencias electrónicas

Desde el punto de vista del proyecto las trazas que registrarán las aplicaciones, se consideran como información susceptible de ser tratadas como evidencias electrónicas.

Desde una perspectiva no forense, la Gestión de la información orientada a la evidencia, se hace necesaria para proporcionar la debida confianza en la transparencia de la gestión y una más eficaz y eficiente respuesta ante incidentes o requerimientos judiciales. Así como una mayor confianza en el resultado de acciones derivadas de actuaciones legales, regulatorias o contractuales.

Así mismo, las organizaciones pueden obtener otros beneficios derivados de la gestión de los documentos electrónicos en cuanto a la eficacia, calidad de gestión y reducción de costes en todos sus procesos, incluidos los administrativos.

Los objetivos que persigue la organización es identificar la información que necesita para sí misma y para respetar las leyes y regulaciones, de tal forma que es capaz de reconstruir las actividades o transacciones que han tenido lugar de forma confiable. No hay duda de su integridad y la autenticidad podrá demostrarse.

Este registro se puede mantener en el tiempo: las cualidades de la accesibilidad, la interpretación la confianza se puede mantener durante el tiempo que el registro sea necesario, quizás permanentemente.

Una vez alcanzado este objetivo, la información resultante retroalimentará a los servicios y procesos de la Organización que podrá favorecer, la mejora continua de los procesos, los objetivos de Negocio … hasta la satisfacción final del cliente.

1.2 Legalidad general forense nacional e internacional

Revisada la legalidad internacional desde el punto de vista de la seguridad y validez de registros digitales, la legalidad nacional de cada país, vemos que la seguridad digital de la jurisprudencia establecida gira alrededor de los certificados digitales.

Igual que una firma electrónica, la robustez de una prueba electrónica dependerá de muchos factores: Orden Jurisdiccional, carga de la prueba, presunciones legales, libre valoración de la prueba.

Fuera del ámbito de la firma digital también está contemplado de forma muy general cómo se debe proceder a la validez de una evidencia de este tipo. Los conceptos que se van a proceder a describir, son tenidos en cuenta (de una forma más o menos detallada) en todos los países que tienen un proceso de validación de firma electrónica en curso, ya finalizado o pendiente de revisión.

Se ha tomado como referencia la ley española para la descripción de los siguientes conceptos, por proximidad y por tener una legalidad ya avanzada en el ámbito mundial.

1.2.1 ¿Qué es el Onus probandi o la carga de la prueba?

La carga de la prueba es la expresión latina del principio jurídico que señala quién tiene la obligación de probar ante los tribunales. Este fundamento radica en un viejo aforismo que expresa “lo normal se presume, lo anormal se prueba”. Por tanto, quien invoca algo que rompe el estado de normalidad, debe probarlo.

¿Cómo opera la carga de la prueba?

Se manifiesta principalmente en la prueba de la existencia de una obligación (que corresponde al acreedor) y en la prueba de extinción de la obligación (que corresponde al deudor). Con la excepción de que en los procesos sobre competencia desleal y sobre publicidad ilícita corresponderá al demandado la carga de la prueba, de la exactitud y veracidad de las indicaciones y manifestaciones realizadas y de los datos materiales que la publicidad exprese, respectivamente.

1.2.2 Presunciones

¿Qué es una presunción legal *iuris et de iure*?

Es aquella que se establece por ley y que no admite prueba en contrario, es decir, no permite probar que el hecho o situación que se presume es falso, a diferencia de las presunciones i*uris tantum* que permiten probar que son erróneas.

¿Qué es una presunción iuris tantum? Es aquella que se establece por ley y que admite prueba en contrario, es decir, permite probar la no existencia del hecho o situación que se presume.

Presunciones asociadas a la firma electrónica. La "firma electrónica avanzada" tiene en relación con un documento electrónico el mismo valor jurídico que la firma manuscrita en relación con los consignados en papel. Por ello es obligatorio su admisión como prueba en juicio, la cual debe ser valorada conforme a los criterios de apreciación judicial establecidos en las normas procesales (es decir, si aquél contra quien se imputa un documento firmado electrónicamente alega error o falsedad, intervienen los peritos y, a la vista de sus dictámenes y de las alegaciones de las partes, decide el juez).

No obstante, existe una presunción legal favorable a la validez de la firma electrónica cuando el Prestador de Servicios de Certificación que ha intervenido en la misma está "acreditado" y el dispositivo de creación de la firma empleado por el firmante está certificado oficialmente.

En caso de la firma electrónica simple o no avanzada sólo se garantiza que no se rechazará de plano su admisión como prueba en juicio por el mero hecho de haberse extendido en forma electrónica.

1.2.3 Fuentes de prueba vs medios de prueba.

Son fuentes de prueba todas aquellas realidades susceptibles de, o bien convencer al juez de una afirmación de hechos realizada por una de las partes en un proceso, de la autoría de unos hechos susceptibles de sanción, o bien de fijar determinados hechos como ciertos: interrogatorio de partes, documentos públicos, documentos privados (el soporte en que se hallen los documentos firmados electrónicamente será admisible como prueba documental en juicio), dictámenes de los Peritos, reconocimiento judicial, interrogatorio de testigos, … y cualquier otro medio que no esté expresamente previsto en los anteriores, de los que pueda obtenerse certeza sobre los hechos relevantes.

Son medios de prueba todo el conjunto de trámites procesales necesarios para introducir válidamente cualquiera de estas realidades en el proceso.

1.2.4 Ciclo de vida de la evidencia electrónica.

La fragilidad de los medios de prueba en soporte digital (que no sólo consiste en su volatilidad sino en la facilidad con la que son reproducibles y falsificables) obliga a plantearse de manera muy sería la protección de la información de los SSII, no sólo en vistas a que esa información sea veraz y oponible a terceros (clientes, trabajadores, outsourcers, etc ….), sino que permita dar una imagen fiel internamente al auditores y controladores de seguridad.

Durante la generación/creación, tratamiento mantenimiento, recogida, captura y almacenamiento del dato-documento electrónico ha de responder a un hecho-acción cierto: no deben de ser datos generados de propósito, aleatoria o automáticamente. Resultan aceptables como hechos ciertos los datos generados por aplicaciones que toman decisiones automáticamente (por ejemplo, determinadas órdenes en mercados continuos de compra-venta cuando se dan parámetros predeterminados) siempre que sea posible establecer como cierta la acción de generación y mantenimiento.

La razonabilidad de certeza de la prueba electrónica que, en su caso, se analice y se aporte en juicio no se consigue sin un entorno de control y la aplicación de estándares.

1.2.4.1 Durante la extracción de la prueba debe tenerse en cuenta.

Obtención lícita de la prueba: no puede obtenerse con infracción legal o de derecho fundamental. Por ejemplo, es ilícita la prueba obtenida de una CPU sin orden de entrada y registro en el domicilio y sin que en la apertura, clonado o volcado del disco duro se encuentre presente el imputado. Genera problemas la entrada no consentida a un ordenador personal por puerta trasera: posible intervención de las comunicaciones.

No alteración y conservación de la prueba. Esto afecta tanto a las medidas técnicas y de seguridad en el momento de la extracción, en el traslado del medio de prueba y de su copia para análisis a un lugar/laboratorio, a la conservación segura del medio de prueba original (de preferencia en un lugar distinto al del lugar en el que se encuentra la copia para análisis).

Las entidades privadas que efectúen la extracción y conserven los medios de prueba:

1. Han de mantener una debida independencia con respecto al dueño de la fuente de prueba.

2. Pueden actuar bajo la forma de testigo perito en el proceso posterior.

3. Puede ser independiente de la entidad que efectúe el informe pericial o que realice la presentación de la fuente de prueba en forma de medio de prueba admisible en juicio.

4. Conviene considerar la conveniencia de mantener ambas funciones (extracción y custodia) como independientes para reforzar la validez posterior de la prueba.

1.2.5 Características de la prueba pericial

Para que una prueba pericial pueda ser tenida en cuenta primeramente deberá estar redactada por escrito que permita su archivo y distribución, y deberá ir acompañada por todo aquel suplemento en forma de documento, instrumento o material adecuado que apoye la explicación y permita exponer el parecer del Perito sobre lo que haya sido objeto de la pericia.

Además, la prueba pericial deberá auxiliar, de forma objetiva e independiente, cualquier duda técnica, no legal, que el Juez o las partes puedan tener. En otro caso, si hay cualquier indicio de duda el perito podrá ser llamado a declarar.

1.3 Cumplimiento de la legislación obligatoria

El Real Decreto 994/1999 de 11 de junio aprueba el Reglamento de medidas de seguridad de los ficheros automatizados que contengan datos de carácter personal, desarrollando los artículos 9 y 44.3h de la Ley Orgánica 15/1999 de Protección de dichos Datos. Y la mejoras incluidas en la ley 59/2003 para incluir todas las premisas y formalizar el uso de certificados digitales.

El Reglamento exige de hecho el escalón más elemental de salvaguardas (tras aplicar el sentido común para activar con eficacia las medidas elementales que suelen venir ya instaladas con los sistemas). Contiene 29 artículos organizados en 6 capítulos y una Disposición transitoria única. Ésta indica los plazos de implantación de las medidas (6, 12 y 24 meses según los niveles los niveles

básico, medio y alto respectivamente, para los sistemas de información en funcionamiento; y 36 meses para adecuar los que *"no permitan tecnológicamente la implantación de alguna de las medidas de seguridad previstas en el presente Reglamento"* (plazo que acaba el 11 de junio del 2002). Estas medidas de seguridad *"se configuran como las **básicas** que han de cumplir todos los ficheros que contengan datos de carácter personal"*.

El ***Capítulo I de Disposiciones generales*** da como objeto del Reglamento *"establecer las medidas de índole técnica y organizativas necesarias para garantizar la seguridad que deben reunir los ficheros automatizados, los **centros de tratamiento, locales, equipos, sistemas, programas y las personas** que intervengan en el tratamiento automatizado de los datos de carácter personal sujetos al régimen de la Ley Orgánica 5/1992"* (ahora la 15/1999). Las ***"medidas de seguridad exigibles",*** válidas para accesos a través de redes de comunicaciones, ficheros temporales y tratamiento fuera de los locales de ubicación (que ha de autorizar expresamente el responsable del fichero), se clasifican en 3 niveles:

- ***Medidas de nivel B básico*** para todos los ficheros que contengan datos de carácter personal.

- ***Medidas de nivel M medio*** (además de las de nivel básico) para *"los ficheros que contengan datos relativos a la comisión de infracciones administrativas o penales, Hacienda Pública, servicios financieros y aquellos ficheros cuyo funcionamiento se rija por el artículo 28 de la Ley Orgánica 5/1992"* ( ) y *"cuando los ficheros contengan un conjunto de datos de carácter personal suficientes que permitan obtener una evaluación de la personalidad del individuo"*.

- ***Medidas de nivel A alto*** (además de las anteriores) para *"los ficheros que contengan datos de ideología, religión, creencias, origen racial, salud o vida sexual así como los que contengan datos recabados para fines policiales sin consentimiento de las personas afectadas"*.

Los **Capítulos II, III y IV detallan las Medidas de seguridad de nivel básico, medio y alto,** empezando por el ***Documento de seguridad,*** que *"deberá mantenerse en todo momento actualizado y deberá ser revisado siempre que se produzcan cambios relevantes en el sistema de información o en la organización del mismo"* y cuyo contenido *"deberá adecuarse, en todo momento, a las disposiciones vigentes en materia de seguridad de los datos de carácter personal"*.

2 DESARROLLO

En este apartado definiremos la situación actual y la deseada para conocer los siguientes pasos a realizar.

2.1.1 Proceso de negocio actual: productos relacionados

A día de hoy, la necesidad de registro de logs se intenta satisfacer, de forma relativamente sencilla, haciendo uso de productos y especificaciones ya existentes (comerciales o no) que, mediante librerías o procesos independientes, se implementan todas las necesidades de configurabilidad de la gestión de mensajes de registro.

Todos ellos implementan primitivas muy similares pero, por arquitectura, se distinguen dos grandes familias de sistemas de logs y ya mencionados anteriormente. Sobre estos modelos será donde nuestro producto deberá hacer notar su valor añadido.

2.1.1.1 Log4j

Log4j es una biblioteca open source desarrollada en Java por la Apache Software Foundation, que permite a los desarrolladores de software elegir la salida y el nivel de granularidad de los mensajes o "logs" en tiempo de ejecución, y no a tiempo de compilación, como es comúnmente realizado, mediante el uso de archivos de configuración externos. Log4J ha sido implementado en otros lenguajes como: C, C++, C#, Perl, Python, Ruby y Eiffel.

Prácticamente todas las opciones de configurabilidad, para un sistema de logs, están implementadas, permitiendo la configuración de Appenders y Layouts que dan la opción de configurar el destino de los mensajes de registro: base de datos, mail, fichero, … u otros destinos a elección del usuario implementados de forma programática. Los Layouts permiten configurar el formato en el que se registra el mensaje de registro, pero no permite la inclusión de campos nuevos, (como en nuestro caso podría convenientemente ser, la de una firma digital si no forman parte del texto asociado al mensaje de registro).

2.1.1.2 Syslog-ng

Syslog-ng es una aplicación de código abierto del Protocolo Syslog para Unix. Sys-logng extiende el modelo original de syslogd con contenido filtrados, ricas capacidades de filtrado, opciones de configuración flexible y agrega características importantes al registro del sistema, como el uso de TCP para el transporte. A día de hoy syslog-ng es desarrollado por Balabit IT Security Ltd. Tiene dos ediciones con un código base común. El primero se llama syslog-ng Open Source Edition (OSE) con la licencia LGPL. El segundo se llama Premium Edition (PE) y tiene plugins adicionales (módulos) bajo licencia propietaria.

Syslog-ng utiliza el protocolo quasi-standard de syslog BSD, especificado en RFC 3164. Como el texto de RFC 3164 es vago, y es sólo una descripción informativa y no una norma, surgieron varias extensiones incompatibles de la misma. Desde la versión 3.0 también soporta el protocolo syslog estándar especificado en RFC 5424, que fue lanzado en 2009. Syslog-ng se esfuerza para interoperar con una amplia variedad de dispositivos, y se puede personalizar el formato de los mensajes de retransmisión.

Las más importantes extensiones del Protocolo original aprobado por syslog-ng:

1. Fecha y hora ISO 8601 con información de detalle y la zona horaria de milisegundo.
2. La adición del nombre de relés en los campos de acogida para que sea posible seguir la trayectoria de un determinado mensaje ha atravesado.

3. Transporte fiable utilizando TCP.

4. Cifrado TLS (desde 3.0.1 en OSE 1).

El protocolo syslog es muy sencillo: existe un servidor de syslog, conocido como syslogd (demonio de syslog que puede estar en una máquina diferente), en el que un cliente envía un pequeño mensaje de texto (de menos de 1024 bytes).

Los mensajes de syslog se suelen enviar vía UDP, por el puerto 514, en formato de texto plano. Algunas implementaciones del servidor, como syslog-ng, permiten usar TCP en vez de UDP, y también ofrecen Stunnel para que los datos viajen cifrados mediante SSL/TLS.

Aunque syslog tiene algunos problemas de seguridad[1], su sencillez ha hecho que muchos dispositivos lo implementen, tanto para enviar como para recibir. Eso hace posible integrar mensajes de varios tipos de sistemas en un solo repositorio central.

2.1.1.3 Comparativa

No se va a poder comparar ambas herramientas para ver cuál es mejor que la otra. Ambas son excelentes y la razón de ser y de elección de ambas herramientas obedecen a necesidades bien distintas. Sólo vamos a poder comparar ambas desde la perspectiva que comulga con nuestro dilema: dar la mayor fuerza legal posible a los registros de logs de un SSII y su rendimiento.

En primera instancia y para syslog-ng este criterio la hace favorita. Los registros de syslog se usan y se han usado durante más de treinta años para la auditoría de seguridad de los SSII. Pero tiene un inconveniente, está basado en un servidor, y existe aún mas experiencia, si cabe, en ataques a sistemas de este tipo.

Por el contrario. Log4j es una librería que debe e integrarse por completo en la misma aplicación que da servicio.

El rendimiento que conseguiríamos basándonos en syslog debería ser mejor, ya que descarga sobre un servidor, la funcionalidad de registró físico, la independencia del entorno (lenguaje de programación, sistema operativo, …) y podríamos usar funcionalidades de seguridad implementadas en versiones como syslog-ng.

Pero deberíamos unir el destino de nuestro SSII al correcto funcionamiento del servidor syslog, por no mencionar los agujeros de seguridad que tiene. La realidad es que el api de las versiones de syslog no ofrece esta posibilidad, es decir, si se envía un registro a syslog puede que se registre o no, si este ha dejado de funcionar por un reincio o cualquier otro tipo de motivo (por ejemplo un ataque), no hace de esta la mejor elección. Además el rendimiento es muy malo comparado con prácticamente cualquier otro sistema de registro de logs.

La opción que nos queda es aprovechar el carácter de código abierto de log4j, mejorarlo en este sentido y hacerlo público. Todos los sabores de esta herramienta que hay en el mercado, son desde el punto de vista de nuestro dilema, candidatos para aportar requisitos de sistema a nuestro proyecto. Ya que nuestra intención es ofrecer un nuevo sabor y competir en este mismo mercado sin obligar a nuestros usuarios a renunciar a las mejores bazas que otras soluciones implementan, si eligen nuestra opción.

[1] Veasé http://datatracker.ietf.org/wg/syslog/

Vamos a remarcar las bazas, requisitos o puntos irrenunciables que nuestro sistema quiere garantizar y veremos como estas son enfocadas por otros productos. Estos criterios están ordenados por importancia y durante el camino de esta comparación veremos que funcionalidades deberemos hacer nuestras, ya que son las que el usuario final necesita, y de las conclusiones obtendremos los requisitos de seguridad y rendimiento, que de implementarse para nuestro proyecto ofrecerán el valor añadido que buscamos.

Los ficheros de logs deben de ser auditados y mantenidos para auditorias

y estudios forenses futuros.

Este es el objetivo principal, nuestro sistema debe de tener un enfoque claramente forense. Este requisito tiene unas implicaciones directas sobre el formato de la información que deberá ser guardada.

Nuestro sistema deberá implementar mecanismo criptográfico que permita al auditor de un mensaje de log determinar la entidad originadora de dicho mensaje (autenticación de origen y no repudio), y confirmar que el mensaje no ha sido alterado desde que fue firmado por el originador (integridad). Es decir tanto el nombre del fichero como la información, que contenga deberán ser suficientes para garantizarlo.

Aproximándonos a esta solución desde Log4j. Este sistema de registro de logs nos ofrece como única opción el uso de la clase Appender.

En particular, la clase derivada FileAppender es un OutputStreamAppender que escribe en el archivo nombrado en el parámetro de nombre de archivo. El FileAppender utiliza un FileManager (que se extiende OutputStreamManager) para llevar a cabo realmente el archivo de E / S. Mientras FileAppenders de diferentes configuraciones no se pueden compartir, los FileManagers puede ser si el gerente es accesible. Por ejemplo, dos aplicaciones web en un contenedor de servlets pueden tener su propia configuración y escribir de forma segura en el mismo archivo si Log4j es en un cargador de clases que es común a los dos.

Observemos una configuración estándar para un FileAppender:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" name="MyApp" packages="">
      <Appenders>
            <File name="MyFile" fileName="logs/app.log">
                  <PatternLayout>
                        <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
                  </PatternLayout>
            </File>
      </Appenders>
      <Loggers>
            <Root level="error">
                  <AppenderRef ref="MyFile"/>
            </Root>
      </Loggers>
```

```
</Configuration>
```

En negrita se han marcado las líneas de interés de la configuración. En File se define el nombre del fichero que solamente puede ser una cadena y sobre la que se puede aplicar una transformación solamente para el rotado de logs si se configura un tiempo o un tamaño de rotación. Este nombre no puede incluir ningún tipo de información "customizable" por el usuario sin cambios de desarrollo en la herramienta. Lo mismo ocurre para la configuración del patrón de impresión de trazas (Pattern) en la que no se puede incluir otra información que no sea la fecha o información similar.

No existe una funcionalidad implementada desde log4j que permita el registro de logs bajo las premisas de este requisito esencial.

Este vacío en la configuración, aunque amplia, hace estéril ahondar más sobre esta cuestión. Y a parte de lo ya indicado no existen de iniciativas, a día de hoy, conocidas de tipo forense alrededor de log4j.

Intentándonos acercar desde syslog-ng debemos hacerlo no desde un punto de vista técnico sino que debemos ver qué problemas a este criterio ofrece esta solución. Como ya hemos insinuado existen problemas de infraestructura que pueden impedir que este sistema sea viable para nuestros propósitos y su estudio no será para fijarnos o usarlo como alternativa sino para, a pesar de su popularidad apartarnos de los errores que, desde el punto de vista forense, comente.

Efectivamente Syslog es intrínsecamente inseguro. Las transmisión de información de Syslog está basada en UDP, el cual es un protocolo sin conexión y por lo tanto poco fiable para fines forenses a menos que LAN separadas se utilicen para la transmisión y la colección de archivos de registro. Las especificaciones del protocolo citan las brechas en la definición de la norma. Aunque algunas de estas deficiencias se subsanan en RFC 3195, el estándar dista ampliamente de lo viable desde el punto de vista forense.

Los principales problemas en el uso de este protocolo se pueden clasificar en tres categorías:

1. **Transmisión de los mensajes.** Como ya se ha apuntado Syslog usa UDP para la transmisión de los mensajes. Esto hace poco fiable la comunicación entre las partes. Algunas implementaciones de Syslog (como la syslog-ng en estudio) permiten elegir el canal de comunicación. Otra solución es usar una conexión punto a punto o una subred dedicada para recoger los logs. En cualquier caso, un hacker con acceso a la red de comunicación entre el origen y el destinatario puede escuchar y borrar los mensajes que detecte de interés. Esto desafortunadamente no puede ser detectado porque no hay "acuse de recibo" o numeración secuencial de los registros.

2. **Integridad de los mensajes.** No existe mecanismo en el protocolo para salvaguardar la integridad del mensaje.

3. **Autenticidad de los mensajes.** No hay verificación de origen de la información, sin más.

Ante esta perspectiva han existido diversas mejoras y versiones con mayor o menor éxito, del protocolo para cubrir estas debilidades, mayormente introduciendo un sistema de criptografía en el canal que aprovechando su funcionalidad resuelvan el problema. Pero una vez identificado el tipo de

tráfico entre el origen y el destino el hacker tiene a su disposición todo un catálogo de vulnerabilidades.

Sea cual sea el sistema elegido, los logs deben guardarse con seguridad para ser utilizados como prueba en la investigación de una violación del sistema. Sin embargo requiere que la prueba, en otras palabras los logs, satisfaga en detalle los siguientes criterios de análisis forense:

1. **Admisibilidad:** Deben ser admisibles como evidencia según la normativa en la sala del Tribunal.

2. **Autenticidad:** Se debe demostrar que los registros contienen evidencia del incidente en cuestión.

3. **Integridad:** Los registros deben representar la historia entera del incidente, no sólo una parte.

4. **Confiabilidad:** No debe de haber duda en cuanto a cómo los datos fueron recolectadas, su autenticidad y exactamente cómo se han custodiado.

5. **Credibilidad:** Deben ser fácilmente entendida y creíble en la sala del Tribunal.

¿Qué implicaciones tiene este requisito en nuestra implementación? Sabiendo que no nos ofrecen las alternativas de mercado y cuales son sus defectos nuestro sistema deberá satisfacer:

1. La información viaja por protocolos con vulnerabilidades conocidas o por descubrir. La mejor opción es que el fichero no viaje, es decir, el fichero deberá ser local al proceso que lo ha generado. Usando llamadas al sistema y destino un fichero.

2. El fichero auditable deberá garantizar, que se registre el orden de registro.

3. Deberá garantizarse el origen del registro. Esto es deberá de garantizarse el proceso que lo generó y este a su vez proviene de un binario que lo hizo posible, deberá estar certificado y esta “certificación” incluida en el fichero de logs.

4. Al igual que su origen la hora de registro deberá estar certificada.

El sistema de registro de logs deberá de ser aplicable a un sistema de alto rendimiento.

Un sistema de alto rendimiento lo define tanto el volumen como el orden de procesamiento paralelo que se requiere para los problemas que resuelve. En este campo entran en juego tecnologías computacionales como los clusters, supercomputadores o mediante el uso de la computación paralela. Y acota, para cada caso, la arquitectura que deberá tenerse en cuenta.

Aplicado a nuestro caso representa que el volumen de E/S sobre el repositorio de logs será abundante, así como el uso de CPU. De lo visto en el requisito anterior ya podemos deducir que será

"stand alone", es decir que la inteligencia del registro de logs estará en la misma ejecución del proceso que se está trazando, lo que implica un "freno" al rendimiento que podremos ofrecer. Por lo tanto la elección de **la implementación deberá ser optima para el tratamiento de E/S**.

Además estos sistemas hacen, frecuentemente y de forma masiva, uso de servicios externos que deberán de tenerse en cuenta. Este hecho introduce los sistemas en la nube que implica diversos retos y problemas de seguridad que hay que analizar para garantizar que nuestra solución es compatible.

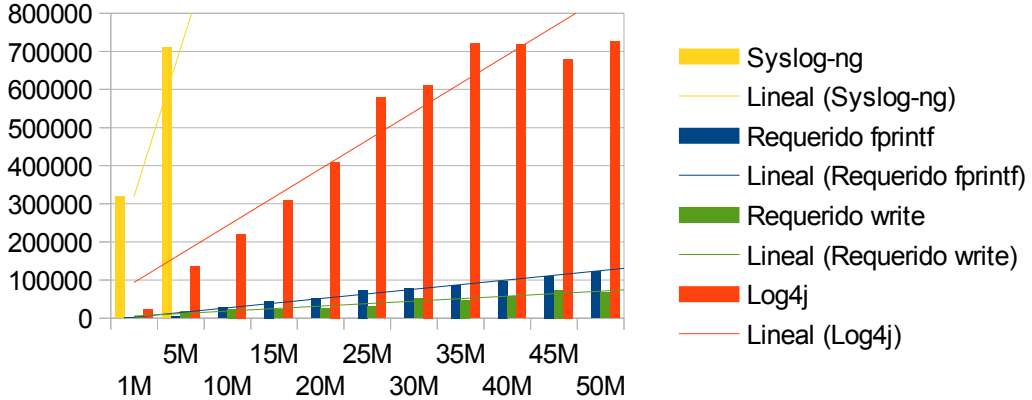
En los últimos años, el cloud computing[2] se ha convertido en popular como un paradigma de computación rentable y eficiente para ofrecer de forma económica una alta disponibilidad y alto rendimiento. Por desgracia, las arquitecturas de computación en nube de hoy no son diseñados para la seguridad y el análisis forense.

En primer lugar, el sistema de almacenamiento ya no es local. Por lo tanto, incluso con una citación judicial, los agentes encargados de hacer cumplir la ley no puede confiscar el ordenador del sospechoso y obtener acceso a la archivos de sospechosos. En segundo lugar, cada servidor de la nube contiene archivos de muchos usuarios. Por lo tanto, no es factible para aprovechar servidores desde un centro de datos sin violar la privacidad de otros muchos usuarios. En tercer lugar, incluso si los datos que pertenecen a un sospechoso en particular se identifican, separarlos de datos de otros usuarios es difícil.

Para tales desafíos, las nubes no pueden ser utilizadas para cuidar de la salud de la seguridad de los datos relacionados con la empresa, o nacionales, que requieren de auditoría y el cumplimiento normativo.

Como resultado, la medicina forense en la nube trae nuevos retos, tanto de nivel técnico y jurídico, que nosotros resolveremos una vez más, mediante técnicas de criptografía internacionalmente reconocidas como el uso de certificados digitales. Es decir, **la comunicación fuera de nuestro sistema solamente podrá ser mediante un sistema que garantice la autenticación mutua** permitiendo así a ambos sistemas, el nuestro y el remoto, realizar sus auditorías pertinentes. Y evitar el uso de recursos externos que así no lo lo requieran para garantizar la trazabilidad del flujo de ejecución de cualquier funcionalidad en la que nuestro sistema participe.

Ni syslog ni log4j han hecho foco en cuestiones similares. Se confía en su configurabilidad para todos los problemas de registro local existentes y la delegación de su procesado en terceras partes en el caso de syslog. Del estudio del rendimiento de ambas herramientas en comparación de lo requerido obtenemos el siguiente gráfico:



[2]Ver Artícu

La prueba se ha realizado sobre una misma traza de 67 caracteres. El eje X indica en millones las trazas que se registran y en el eje Y el tiempo en milisegundos que han tardado en grabarse. Las características del equipo en el que se han realizado las pruebas son:

Sistema Operativo: Fedora 20
CPU Intel(R) Core ™ i5-2450M CPU @ 2.5 Ghz
Memoria física: 6144 MB RAM
HDD: TOSHIBA MK7575GSX. Buffer: 8192KB Velocidad: 5400 RPM

Donde los datos para Requerido representa la carga útil que debe asumir el sistema para resolver el registro de logs . Existen dos "Requerido" una mediante el uso de *frprintf* y una segunda que mide el tiempo basado en una escritura directa con *write* más la latencia que introduce una instrucción *printf* que en cualquier caso necesitaremos introducir para registrar todo registro. Y de la que podemos deducir lo siguiente:

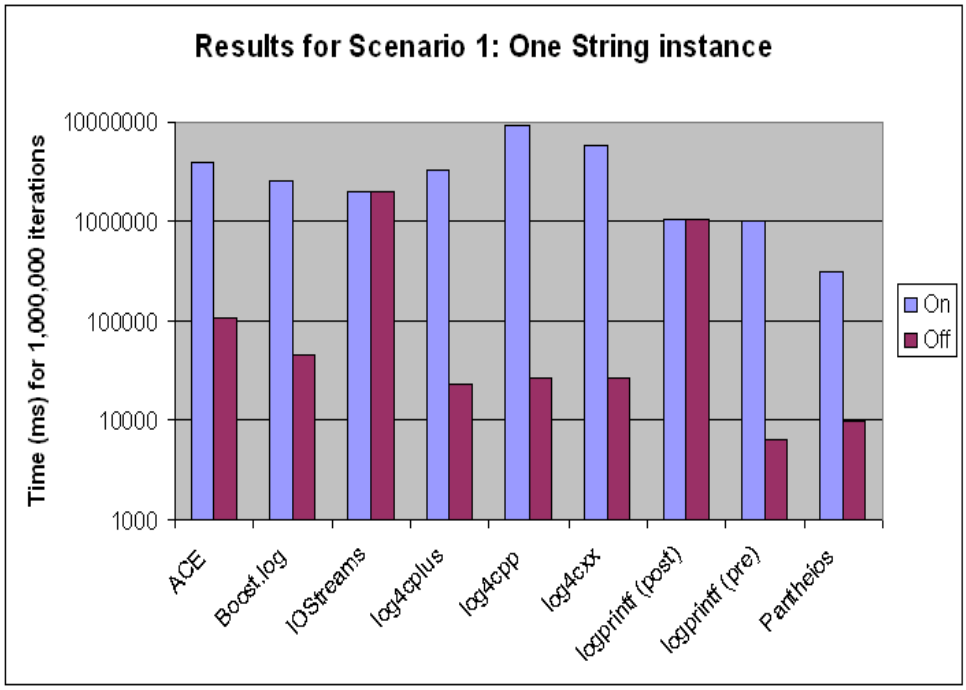
- *fprintf* promete hacer una llamada al sistema (*write)* por cada acumulado de bytes pendientes de grabar igual al tamaño de un bloque para mejorar el rendimiento. Pero de la gráfica se deduce que esto no es necesariamente cierto según el compilador y tamaño (y tipo) de buffer configurado.

- Log4j tiene un rendimiento malo en comparación con la tarea útil que debe realizar.

- Para Syslog-ng, a pesar de consistir en enviar un "servidor de logs" la responsabilidad de registro, esta concurrencia, no impide que el rendimiento que ofrece lo haga inviable para registro masivo de logs. Las llamadas a sistema para la apertura de sockets tipo AF_UNIX, por cada llamada penalizan de esta forma el rendimiento.

  Al margen de log4j y syslog existen un sinfín de herramientas con características muy similares pero distinto rendimiento. Podemos tomar como referencia las mismas elegidas por Pantheios para probar la mejora que introduce a propósito del rendimiento: ACE, Boost.log, log4cpp, log4cplus, log4cxx en su página web[3]. En el primero de sus escenarios de prueba, (con un hardware similar al anterior y para el registro de un string de tamaño de 67 bytes), podemos observar esta gráfica (el resto de los escenarios informan de tiempos similares):

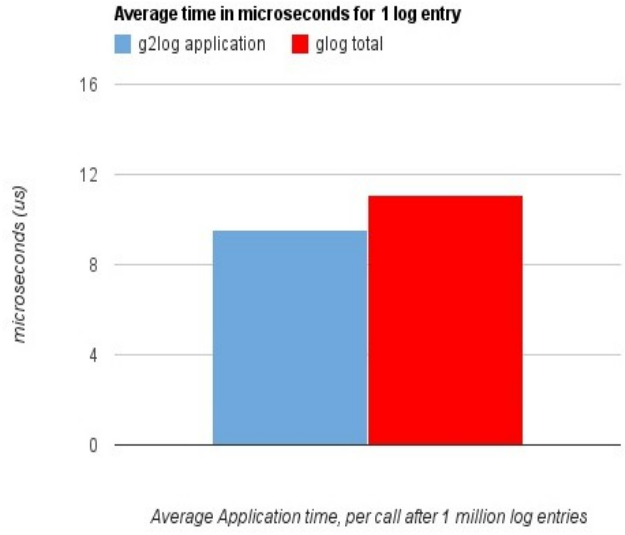
[3] http://www.pantheios.org/performance.html



Los tiempos están muy por encima de la prueba somera que hicimos con anterioridad para ver que requisitos de sistema necesitamos para grabar en disco comparado con lo que nos ofrece log4j aún tomando los tiempos OFF (tiempo en el que la traza no se registra por no cumplir la condición de nivel de log). El motivo es la compatibilidad con threads que se incluye en todas estas herramientas.

Esta compatibilidad implica que, a parte de un uso masivo de I/O sobre disco, existe el retraso por sincronización y serialización de los registros que dispara el tiempo y puede bloquear los threads implicados en operaciones de registro.

Otras herramientas ofrecen tiempos mucho mejores como pueden ser g2log y glog. Los tiempos se aproximan mucho más (llegándolos a mejorar) a los medidos anteriormente. Los tiempos ofrecidos por estas son:



Average Application time, per call after 1 million log entries

Estos tiempos son muy similares a las llamadas que hacen registros directos sobre un fichero. ¿Cómo puede ser? Usan un registro asíncrono en disco, es decir una instrucción de registro no escribe en disco sino que pasa a una cola que se encarga de realizar este registro. Pero, ¿No hace fprintf algo similar? Efectivamente, fprintf realiza algo similar, minimizando las llamadas al sistema para mejorar el rendimiento, grabando en disco hasta tener completo el buffer con tamaño igual al bloque de disco que hace que sea óptimo. Simplemente usan, no un buffer, sino una cola de peticiones de registro con mucha mayor capacidad (y embebido en el ejecutable que está corriendo) y usan técnicas de formateado ultrarápidas de cadenas como FastFormat.

En realidad estas técnicas de formateado son en cualquier caso mas rápidas que cualquier otra técnica de formateado excepto sprintf y el encolado implica que no hay garantías ante un fallo y un gran número de registros no den cuenta de lo que realmente se ha ejecutado.

En definitiva, en cuanto a rendimiento, para las opciones de registro existentes, antes de usarlas hay que leer la letra pequeña y su carácter de "propósito general" las hacen ineficientes o no viables para según que propósito.

Deberá satisfacer la legalidad vigente[4].

El primer requisito que implica que los registros de logs deben de ser auditables a efectos forenses marca un criterio objetivo sobre la fortaleza de una evidencia. También conocemos la legalidad general forense aplicable en el ámbito internacional y es la que mide la validez a la evidencia. Es un buen punto de partida ya que el responsable de la seguridad del sistema deberá convencer al público implicado que la información que ofrece para la auditoría es fiable por lo que debemos demostrar control y mantenernos cercanos a las recomendaciones. Destacan en este ámbito las técnicas y mecanismos criptográficos que tienen un protagonismo singular y creciente en la garantía de dimensiones de la seguridad tales como la confidencialidad, la integridad y la autenticación; forman parte de estructuras más complejas como servicios, aplicaciones e infraestructuras; y, de hecho, se han convertido en componentes clave sobre los que se sustentan piezas esenciales para los servicios de administración electrónica como puede ser la firma electrónica y el fechado electrónico. En el campo de las técnicas y mecanismos criptográficos, las normas producidas por el SC27 ofrecen especificaciones relativas a cuestiones tales como las siguientes:

- Servicios de fechado electrónico (ISO/IEC 18014 Time Stamping Services). Se trata de una norma multiparte que, entre otras cuestiones, describe modelos, servicios y protocolos para el fechado electrónico.

- Algoritmos de cifrado simétricos, de bloque, de flujo y asimétricos (ISO/IEC 18033 Encryption algorithms).

- Funciones hash criptográficas (ISO/IEC 10118 Hash functions).

[4]Ver "Normalización en la Seguridad de los Sistemas de Información" del Ministerio de Administraciones Públicas.

- Esquemas de firma digital que incorporan funcionalidades de autenticación e integridad (ISO/IEC 9796 Digital signature schemes giving message recovery).

- Autenticación de entidades (ISO/IEC 9798 Entity authentication); se trata de una norma multiparte que especifica diversos mecanismos de autenticación, utilizando, entre otras, técnicas de criptografía simétrica y asimétrica.

- Técnicas criptográficas basadas en curvas elípticas (ISO/IEC 15946 Cryptographic techniques based on elliptic curves); en particular, contienen especificaciones para sistemas de firma digital basados en curvas elípticas.

- Requisitos de módulos criptográficos.

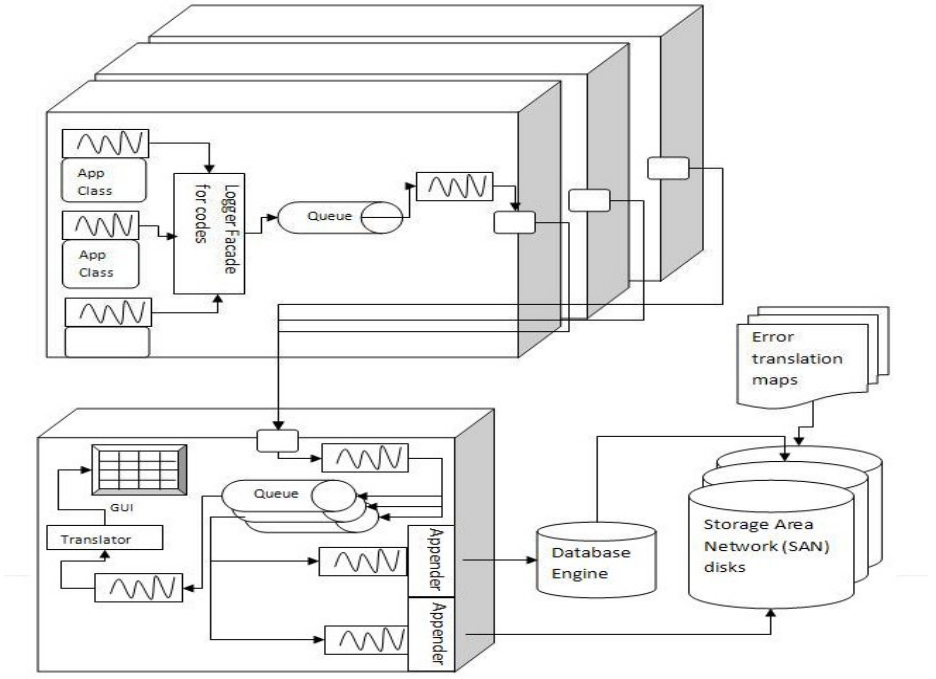
- Gestión de claves (ISO/IEC 11770 Key Management). Trata de conceptos, modelos, servicios y mecanismos para la gestión de claves.

- Mecanismos de no repudio (ISO/IEC 13888 Non repudiation). Trata de modelos, mecanismos y de la utilización de técnicas de criptografía asimétrica para el no repudio.

Los usuarios de productos y sistemas de tecnologías de la información necesitan confianza en que las soluciones que puedan desplegar satisfacen los objetivos de seguridad declarados por las mismas, en el sentido de que son capaces de hacer frente a las amenazas identificadas, de satisfacer las políticas de seguridad definidas en el marco de unos determinados supuestos o hipótesis de seguridad, así como de incorporar unos determinados requisitos funcionales y de aseguramiento que contribuyen a la satisfacción de los citados objetivos.

Desde el punto de vista de syslog como de log4j no hacen referencia a estos mecanismos de control. Si bien son altamente configurable y flexibles para adaptarse a estas recomendaciones no ofrecen, por ejemplo, tamaño suficiente para un registro de log que puede ser necesario para dejar constancia de información de control que permita su auditoría en el caso de syslog, que no se recomienda mas de 80 bytes para un mensaje. En el caso de log4j existen aproximaciones de appenders y versiones que han intentado estandarizar el uso para fines de auditorías pero no son capaces, por ejemplo, de garantizar que el orden de registro se ha respetado, sin manipulación, lo que cuestiona el valor de las evidencias que registra.

Nuestro sistema será compatible con las recomendaciones de uso, ley y estándares actualmente en vigor. Para conseguir esto cada entidad que lo use deberá ser capaz de acceder a una configurabilidad o adaptación de uso tal que le permita elegir como aproximarse a dichas recomendaciones, al menos en lo que atañe a la recolección de logs de sus sistemas. El sistema de registro será compatible con el siguiente esquema:



Nótese el módulo "Logger Facade for codes" el cual será el interfaz con el desarrollo y puerta de entrada a la custodia de los registros.

El sistema elegido deberá ser de código abierto.

En este punto, ni syslog ni lo4j ofrecen ningún tipo de inconveniente. Ambos sistemas son de código abierto sin duda clave del éxito que han tenido. Nuestro nuestro sistema deberá de continuar con este modelo de explotación por varios motivos:

1. No se pretende conseguir un estándar, pero sí maximizar la penetración en el mercado.

2. En seguridad algo que no sea de código abierto causa desconfianza. Sobre todo si es nuevo.

3. Posibilidad de ofrecer extensiones de adaptación a las circunstancias de cada SSII.

El resultado del uso de nuestra herramienta deberá permitir el procesado automatizado de terceras partes. Es decir, desarrollaremos, por requisito de negocio y siguiendo el espíritu de código abierto, solamente aquello que resuelva el problema en cuestión y que genere información (y binarios en la medida de lo posible) estandarizada que permita su integración en un stack de herramientas ("Menos es mas").

2.2 Proceso de negocio requerido

De las necesidades descritas y de las conclusiones de la situación actual se hace patente la construcción de esta nueva herramienta que supla las carencias existentes. Podemos concluir que la

situación actual no se acercan de forma correcta a la necesidad en estudio, por lo que será necesaria la implementación de un sistema que la cubra y se haga un hueco en el mercado correspondiente.

2.2.1 Descripción del proceso de negocio

DESCRIPCIÓN DE LA NECESIDAD	
Nuestro sistema deberá cubrir la funcionalidad de los sistemas de log requerido para un SSII tomando como referencia el ya descrito.	
DESGLOSE DE REQUISITOS DE NEGOCIO	
COD. RN	**DESCRIPCION**
RN1.1	**CONFIGURABILIDAD** Se centralizará sobre uno o varios ficheros toda la configurabilidad posible para el sistema de logs: ruta de fichero logs, nivel de trazas, formato de la línea … tómese como referencia log4j. Se deja a criterio de la solución técnica la granularidad de esta configurabilidad, y su momento de aplicación (ejecución, compilación o implantación) en aras del rendimiento.
RN1.2	**INFORMACIÓN FORENSE** Se incluirá información cifrada que garantice la autenticación de origen, no repudio e integridad de la información que el fichero de log contiene. Así como el orden y hora de registro. Esta información podrá ser validada por una auditoría forense.
DESGLOSE DE REQUISITOS DE PRODUCCIÓN	
COD. RN	**DESCRIPCION**
RP1.3	**ANÁLISIS AUTOMATIZADO DE LOGS** El formato de las trazas deberán de permitir la automatización de alarmas relacionadas con el servicio o la validación de integridad de la información. Y por supuesto la explotabilidad del sistema.
RP1.4	**CUSTODIA DE LOGS** Queda a responsabilidad de Producción la custodia de los logs generados y que deberán ser acordes al estándar que se adscriba. Y será responsabilidad de Desarrollo el facilitar, mediante la configurabilidad que debe tener, la adaptación al estándar que Producción elija.
DESGLOSE DE REQUISITOS DE SEGURIDAD	

COD. RN	DESCRIPCION
RS1.5	**Gestión Infraestructura de clave pública** La seguridad de la información cifrada tendrá como origen los certificados digitales que seguridad proporcione. Queda a responsabilidad de Seguridad el mantenimiento del sistema PKI corporativo. En la medida de lo posible, la implementación ofrecerá la posibilidad de la configuración de sistemas de registro de información forense con clave no centralizada.

DESCRIPCIÓN DE LA NECESIDAD	
El sistema de registro de logs deberá de ser aplicable a un sistema de alto rendimiento.	
DESGLOSE DE REQUISITOS DE NEGOCIO	
COD. RN	**DESCRIPCION**
RN2.1	**SISTEMA EMBEBIDO** La herramienta será embebida en la aplicación a la que da soporte para sus registros de logs. Se renuncia a soluciones cliente/servidor puras.
RN2.2	**SISTEMA THREADSAFE** El sistema implementado será threadsafe por lo que no se incurrirán en problemas de carrera en el uso de las primitivas de registro de logs ofrecidas por la herramienta solicitada.
RN2.3	**“AS FAST AS HELL”** El rendimiento deberá ser un criterio decisivo a la hora de definir la solución técnica para la implementación de la herramienta.
DESGLOSE DE REQUISITOS DE FORMACIÓN	
COD. RN	**DESCRIPCION**
RF2.4	**BUENAS PRÁCTICAS** Se proporcionará a Desarrollo una descripción de buenas prácticas y toda aquella información que ayude a tunear la configuración de registro y proporcione el máximo rendimiento.

RF2.5	**SIMULACIÓN Y PRUEBAS** Se proporcionará el software necesario para las pruebas requeridas que demuestren su correcto funcionamiento. Así mismo este software será un ejemplo de uso de la herramienta que permita su aprendizaje.

DESCRIPCIÓN DE LA NECESIDAD	
El sistema elegido deberá ser de código abierto.	
DESGLOSE DE REQUISITOS DE NEGOCIO	
COD. RN	**DESCRIPCION**
RN3.1	**REPOSITORIO PÚBLICO** Deberá de crearse un repositorio público para la distribución e implantación en otros entornos tanto para desarrollo como explotación.
RN3.3	**"MENOS ES MAS"** Los clientes saben que las herramientas de código abierto ofrecen la posibilidad real de adaptación a sus SSII. Y este enfoque, por parte del cliente, es a su vez el modelo de negocio que ofrecerá nuestro sistema. Esto no significa que el resultado final sea algo inacabado, al contrario debe ofrecer una funcionalidad interesante que capte la atención del cliente y que a su vez quede patente que no es algo cerrado y que puede ofrecer todas mejoras que el usuario final necesite. La solución técnica debe encontrar este punto de compromiso, según las posibilidades de implementación que de forma natural hagan esta oferta.
DESGLOSE DE REQUISITOS DE ARQUITECTURA	
COD. RN	**DESCRIPCION**
RA3.4	**IMPLANTACIÓN** Queda a responsabilidad de Arquitectura el ofrecer los mecanismos oportunos para la implantación del software desplegado en el repositorio público y de su certificación. Desarrollo ofrecerá un control de versiones git que permita su implantación en estos términos.

DESCRIPCIÓN DE LA NECESIDAD	
Deberá satisfacer la legalidad y estándares vigentes.	
DESGLOSE DE REQUISITOS DE NEGOCIO	
COD. RN	**DESCRIPCION**
RN4.1	**LEY 59/2003**[5] Los registros de logs de nuestro sistema deberán poderse amparar en esta ley para demostrar su validez legal y poderse ofrecer como evidencia ante un Tribunal.
RN4.2	**ESTÁNDARES**[6] Nuestro sistema no deberá entorpecer (en todo caso lo contrario) al Responsable de Seguridad en sus actuaciones/decisiones para con la legalidad general actual (es decir, deberá hacer patente una fortaleza defendible y que no genere dudas en un Tribunal) y deberá demostrar control y estandarización. En particular con ISO JTC1 SC27, que es la norma de referencia.

2.3 Casos de uso del Proyecto

2.3.1 Actores

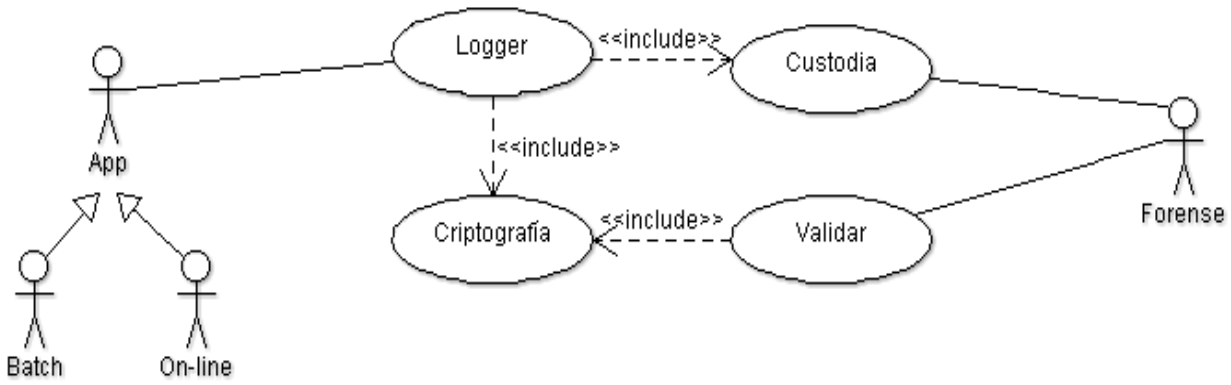
Actor	Descripción
App	Aplicación software que hará uso del registro de logs
Batch	Aplicación de tipo batch
On-line	Aplicación On-Line
Forense	Analista forense de los registros de log de App

2.3.2 Funcionalidades principales del proyecto

En este apartado se incluye un Modelo de Casos de Uso que muestre las principales áreas funcionales a las que afecta el proyecto y su relación con los actores identificados.

[5]Ver texto ley en el BOE ref. BOE-A-2003-23399

[6] Referencia al Estandar ISO JTC1 SC27



2.3.3 Criterios de aceptación del requisito de negocio

De los requisitos de negocio anteriormente expuestos se pueden deducir las funcionalidades que deberá cumplir nuestro producto.

LISTA DE REQUISITOS FUNCIONALES Y CRITERIOS DE ACEPTACIÓN				
COD RF	**TÍTULO DEL REQUISITO**	**DESCRIPCIÓN DEL REQUISITO**	**COD CA**	**DESCRIPCIÓN DEL CRITERIO DE ACEPTACIÓN**
RF1	Configurabilidad	Alta configurabilidad	CA1	Al menos igualarse todas las posibilidades de configurabilidad que no supongan un perjuicio al rendimiento.
RF2	Rendimiento	"As fast as hell"	CA2	Todas las medidas por encima de las posibilidades del sistema rendimiento podrán ser justificadas.
RF3	Confiabilidad	Validación de integridad	CA4	Los ficheros de logs podrán ser validados mediante una herramienta construida para tal efecto.
RF4	Despliegue	Implantación de software	CA5	El software entregado podrá implantarse en entornos pre-productivos/productivos para su evaluación y explotación.
RF5	Automatización	Análisis online de logs	CA6	Los registros de logs serán compatibles con sistemas de detección de alertas.

3 IDENTIFICACIÓN DE LA SOLUCIÓN TÉCNICA EN CADA SISTEMA

3.1 Requisitos Identificados

3.1.1 Ficheros de configuración de log4legal

Deberán existir sendos ficheros de configuración que permitan incluir la información relativa al comportamiento de log4legal durante la compilación y ejecución. Pudiéndose modificar en cualquier momento sin merma del rendimiento o integridad.

Estas variables de configuración deberán tener formato estándar *key=valor*.

El nombre del fichero de configuración podrá venir especificado en tiempo de compilación mediante la definición de una variable de entorno que informe de su localización en el sistema de ficheros.

Los nombres y objeto de los parámetros de configuración serán:

- **log4legal.appender.file.name.ConversionPattern (compilación):** Variable que define el nombre del fichero que se generará mediante una máscara.
- **log4legal.appender.default (compilación):** Instrucción de registro físico del log. La configuración de esta variable debe de permitir usar métodos de registro diferentes a *write*, implementados por el usuario. Por defecto valdrá *write (para un registro directo en el sistema de ficheros)*, en caso de incluir otro valor deberá corresponder con una función implementada por el usuario que admita los mismos parámetros que la instrucción write y a partir de los cuales proceder al registro. Por ejemplo, una función que envíe mediante mensajes POSIX la información de registro a un segundo proceso que se encargue del registro físico. Mediante variables tipo static deberá resolver el resto de parámetros que requiere este sistema de mensajería.
- **log4legal.appender.file.path (compilación):** Variable de entorno que informa de la ruta en el sistema de ficheros para depositar los ficheros de log. Se implementa mediante una variable de entorno para que pueda ser reutilizada por configuraciones distintas.
- **log4legal.level (ejecución):** Nivel de trazas.
- **log4legal.rotation.size (ejecución):** Rotación de logs en caso de que el fichero supere el valor configurado.
- **log4legal.rotation.seconds (ejecución):** Rotación de logs en el caso de que el proceso supere el valor configurado para un mismo fichero de logs.
- **log4legal.timestamp.protocol (compilación):** Método para conseguir la marca de tiempo que dará validez a la firma electrónica.
- **log4legal.timestamp.url (compilación):** Datos de conexión para conseguir el timestamp según el método configurado.
- **log4legal.pki.certificate (compilación):** Certificado digital asignado al proceso.
- **log4legal.pki.hash.method:** Método de hash para enlazar trazas.
- **log4legal.appender.ins.MSJ-Comentar.layout.ConversionPattern (compilación):** Definición del formato de la línea de registro para la instrucción de log *MSJ-Comentar* disponible para el desarrollo del fuente del proceso.

- **log4legal.appender.ins.PRO-Error.layout.ConversionPattern (compilación):** igual que el anterior pero para PRO-Error.

Todas las variables modificables en tiempo de ejecución deberán aplicarse durante el rotado ya que, si las trazas son abundantes la aplicación del cambio será rápida en caso contrario la necesidad no es acuciante.

Los valores de compilación/ejecución determinan en qué momento deberá ser efectiva dicha configuración. Y el criterio que se toma para tomar esta decisión es, no solo el rendimiento sino el grado de impacto en la seguridad del registro. Es decir, un cambio de certificado para la firma implica que en un rotado de logs y hasta el momento del rotado la firma garantizaba un firmante y posteriormente a otro distinto. Estos valores tomados en tiempo de compilación implican la recompilación de los fuentes del proceso y rearranque del mismo en las mejores condiciones para su integridad.

3.1.2 Información de registro

El código generado deberá asegurar la información forense ya mencionada para garantizar RN1.3.

El formato de la configuración de una instrucción *ins* tendra un formato en la que deberán poderse introducir etiquetas del tipo {tipo,valor} que en tiempo de compilación podrán interpretarse para que la instrucción asociada genere una cadena de caracteres formateada con la información deseada. Así pues una configuración del tipo:

{%-5s,hash} {%ld,time(NULL)} MSJ-Comentar [{%s,__FILE__}:{%d,__LINE__}]:{%s,M}\n

Habrá generado una instrucción con estos parámetros, similar a:

*L4L_MSJ_Comentar(int L, char *M);*

Donde L será el nivel de log con el que se quiere registrar la traza y M es una cadena de caracteres con el valor del mensaje a registrar.

Y que al usarla en el código fuente del proceso generará un registro de log con la información similar a:

AC4B3A4B39 15509034 MSJ-Comentar [main.c:25] ¡Hola Mundo!

Es decir %-5s indica el tipo (formato *printf*) y hash es el valor y cualquier instrucción que pueda ser reemplazada en tiempo de compilación como se ha hecho para incluir *time(NULL)*.

hash es el único valor predefinido que se puede usar en estas cadenas de configuración. Esta es una recomendación de configuración pero la funcionalidad deberá ser al menos tan versátil como la descrita.

3.1.3 Inicio y fin de logs.

Antes de usar las instrucciones de registro deberá de ejecutarse, en la función principal, la instrucción *L4L_Init()* que realizará las tareas:

- Lectura de la configuración modificable en tiempo de ejecución y no incluida durante la compilación.

- Calculo y apertura del fichero de logs

- Primera traza con la inclusión de la información firmada relativa a: binario del proceso + marca de tiempo + parámetros. Y a partir del certificado indicado en la configuración.

De forma análoga en la ejecución del fichero *L4L_End()* se incluirá la traza de finalización y el cierre del fichero de logs.

3.1.4 Simulación y pruebas.

Para las pruebas y formación sobre el software construido es necesario facilitar una herramienta que use de forma masiva y con diferentes configuraciones disponibles, las distintas posibilidades que ofrece nuestro producto y la medida de su rendimiento.

3.1.5 Formato librería para configuración por defecto.

Se han descrito, en apartados anteriores, la configurabilidad, compatible con *make*, para la generación de fuentes que se puedan usar en el desarrollo de los procesos que integrarán esta funcionalidad en sus sistema.

Se entregará la configuración correspondiente a la funcionalidad descrita en forma de librería dinámica que permita su inmediata interacción.

4 PROYECTO GIT Y MEDIDAS

El proyecto con el desarrollo de los fuentes que cumplen con los requisitos del apartado anterior puede encontrarse como un desarrollo de código abierto en el proyecto:

https://github.com/jaortizb/Log4Legal.git
o
git@github.com:jaortizb/Log4Legal.git

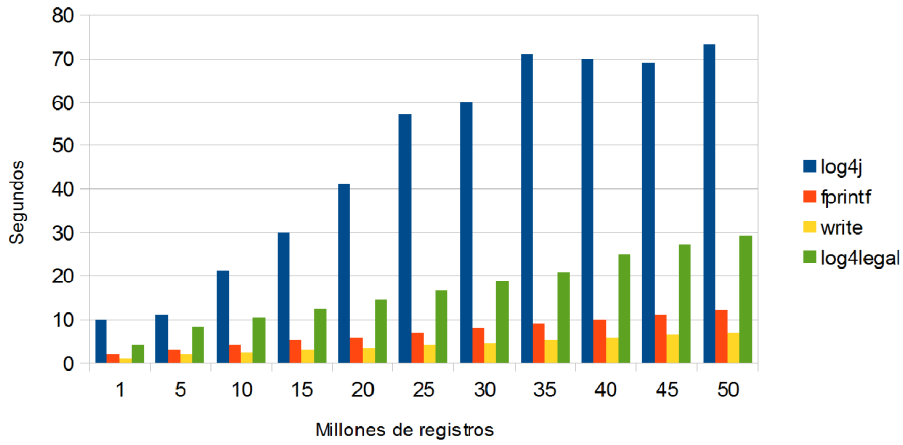
Aquí podrá encontrar los detalles relativos a la implementación, licencia, instalación y manual de usuario completo de la herramienta.

El ejemplo de uso elegido para documentar el rendimiento de nuestra herramienta es:

```
...
#include "l4l.h"

int main(int argc, char **argv) {
double t_begin, t_end;
int i=0,rc;
FILE *FD;

rc = l4l_init("log.cfg","test",argc,argv);
t_begin = get_process_time();
```

```
for(i=0;i<1000000*atoi(argv[1]);i++)
      l4l_printf(0xFFFFFFFF,L4L_MSJ_Comentar,"Hello World!");
t_end = get_process_time();
printf( "\nElapsed time(%d): %.3f seconds\n", i,t_end - t_begin);
l4l_end();

return(0);
}
```

Para este código la medida de rendimiento, en comparación (con el mismo hardware y configuración equivalente) con las tomadas con anterioridad es la siguiente:



Puede verse, que los valores de nuestro sistema de registro son más proximos a los del printf, es decir, el retraso introducido por nuestro sistema de logs es mínimo frente a lo esperado y funcionará correctamente podrá llevarse a entornos de altas prestaciones con un un comportamiento óptimo.

La información de registro generada para el ejemplo indicado y separada por secciones[7] de interés:

1º Información de Licencia de Uso:

```
ZUMS-Version: L4L v.0.3 Copyright (c) 2015 Jose Antonio Ortiz <jaortizb@gmail.com>
MSJ-License: This software is released under the LGPL 2.1 license, see the LICENSE file.
```

2º Sección con la información de arranque y configuración.

```
MSJ-Log: Log File Name /usr/ESORTIZBASJ/l4l/test/repository/test7_20150408_135423_5512.log
MSJ-Bin: Binary file >./a<
MSJ-Def: L4L_ROTATION: YES
MSJ-Def: L4L_HASH_METHOD: sha1
MSJ-Cfg: print_mask=0xffffffff, hash_mask=0xffffffff, sync_mask=0x0,flush_mask=0x0
MSJ-Cfg: buffsize=4098, bufftype=_IOFBF, max_size=5000000, max_seconds=3600
```

3º Línea de firma del arranque. Para este ejemplo la firma ha sido realizada a través de transacciones Bitcoin usado para la implementación del modelo descentralizado de autenticación.

```
MSJ-Timestamp: {
       "hash_sha256":"565d44dd67147a8c9c61bfc47cecb99bf05d2d8ce1cca96f4c82fa5db1bf7c"
      ,"created_at":"2015-04-05T14:09:14.435Z"
      ,"updated_at":"2015-04-05T14:09:14.435Z"
```

[7] En la información descrita en los apartados 3º y 5º se han incluido saltos de línea y tabuladores para mejorar su claridad.

```
       ,"submitted_at":null
       ,"title":"L4L_init/6965db86eec9d07b6604e8613506f29ccfca4e51"
}
```

4º Líneas de registro incluyendo el hash (sha1) encadenado al principio de cada línea.

```
08de115fa2 ffffffff 1428494085 [l4l.c:37] MSJ-Comentar: Hello World!
3b99b6e777 ffffffff 1428494085 [l4l.c:37] MSJ-Comentar: Hello World!
6f34d11023 ffffffff 1428494085 [l4l.c:37] MSJ-Comentar: Hello World!
...
```

5º Línea de firma del contenido del fichero.

```
08e59e51ff ffffffff 1428494097 [l4l.c:41] MSJ-Timestamp: {
        "hash_sha256":"cfabe9ae6dca88f042214c46e9bddbe7228beaf835326966e6911a494196d2f"
       ,"created_at":"2015-04-06T14:11:58.117Z"
       ,"updated_at":"2015-04-06T14:11:58.117Z"
       ,"submitted_at":null
       ,"title":"L4L_end/0704e86ec9d3506f29ccf965db66ca16b86e4e51"
}
```

5 BIBLIOGRAFÍA Y REFERENCIAS

5.1 Libros

"Seguridad en las Comunicaciones y en la Información", G. Díaz Orueta y otros. Ed. UNED. 2004
"Software Libre. Herramientas de Seguridad". Tony Howlet. Ed. Anaya Multimedia. 2005
"Hackers 2". J. Scambray, S. McClure y G. Kurtz. ED. Osborne-McGraw-Hill. 2001
"Extreme Exploits". V. Oppleman, Brett Watson. Ed. Anaya Multimedia. 2006

5.2 Referencias

"Computer Security Incident Handlint Guide". T. Grance. NIST-USA. 2002
"GIAC Security Essentials, Practical Assignment" Version 1.4b. Tan Koon Yaw. SANS Institute 2003.
"Forensic Examination of Digital Evidence: A Guide for Law Enforcement". John Shcroft. U.S. Dep. Of Justice, Apr. 2004
"Helix 1.7 fro Beginners". BJ Gleason and Drew Fahey. Manual Version Mar. 2006
"First Responder's Manual". U.S. Dep. Of Energy Computer Forensic Laboratory. 1999.
"Análisis Forense Digital GNU/Linux". David Ditrich y Ervin Sarkisov. 2002
"Análisis técnicos del Reto Forense ediciones 1 y 2 (comunidad RedIris). 2004 y 2005
"Una Propuesta Metodológica y su Aplicación en The Sleuth Kit y EnCase: Descargar en disco". Octubre de 2005. Jonathan Córdoba; Ricardo Laverde; Diego Ortiz; Diana Puentes.

5.3 URL

www.auditoresdesistemas.com	www.enfsi.org
www.google.es	www.red.es
www.criptored.upm.es	www.forensics-es.org
www.ioce.org	www.securityfocus.com
www.dfrws.org	www.foundstone.com
www.isaca.org	www.opensourceforensics.org

www.iso.org www.e-fense.com	www.boe.es www.oas.org/juridico/spanish/cyb_analisis_foren.pdf

6 ANEXO I. EJEMPLO DE USO

A continuación, usando el registro de logs que hemos construido, vamos a implementar de la forma mas sencilla posible, un ingenio capaz de registrar de forma "certificada" todos los comandos que ejecuta un usuario en su cuenta *linux* a través de una sesión *bash*. Existe una serie de comandos para la gestión de la historia de comandos ejecutados, como por ejemplo el comando *history* que permite eliminar registros y por lo tanto los rastros que un ataque pueda dejar. No se profundizará en la necesidad de esta funcionalidad, se ofrece solamente para fines didácticos. Se incluirán al final las consideraciones pertinentes sobre seguridad a propósito de este ejmplo de uso.

Básicamente, vamos a aprovechar la variable de entorno de *bash* llamada *PROMPT_COMMAND* que permite ejecutar un comando por cada orden que el usuario ejecuta en su sesión bash y usarla para registrar esta línea a parte del típico fichero *.bash_history,* en un fichero *.bash_history_<fecha>_<pid>.log* . El usuario root levantará un proceso que procederá al registro en este fichero y cada usuario que use esta funcionalidad, enviará a este proceso la solicitud de registro, por cada comando que ejecute que procederá a su registro. Comencemos por el fichero de configuración de runtime *log.cfg*:

```
log4legal.test.print_mask=0xffffffff
log4legal.test.buffsize=1024
log4legal.test.bufftype=_IOFBF
log4legal.test.pki.hash.method=sha256
```

Para la configuración en tiempo de compilación modificaremos el fichero *compcfg.h*:

```
#define L4L_MAX_TRZ_LEN 1024 // Máxima longitud de línea
#define L4L_READ_KEY getenv(L4L_KEY) // Password para hash encadenado, el fichero
que contendrá tal valor será pasado por parámetro, dicho fichero será de solo
lectura y solo para el usuario root

#define L4L_MAX_HASH_LEN 10       // Longitud del hash encadenado
#define L4L_TIMESTAMP YES         // Incluir timestamp
#define L4L_HASH_METHOD "sha1"    // Método de hash

#define L4L_FILE_NAME "/var/log/.bash_history_%s_%ld.log",l4l_now("%Y%m%d_%H%M
%S"),getpid() //Nombre del fichero

#define L4L_MSJ_Comentar ">%s< MSJ-Comentar: %%s",L4L_HASH
```

El fuente (test.c) que realizará el registro y que se compilará con la configuración anterior será:

```
int main(int argc, char **argv) {
       int i=0,rc;
       char linea[L4L_MAX_TRZ_LEN]="",L2[L4L_MAX_TRZ_LEN]="";
       FILE * fp;

       rc = l4l_init("log.cfg","test",argc,argv);
       fp = fopen (argv[1], "r");
       fscanf(fp, "%[^\n]",linea);fclose(fp);
       setenv("L4L_KEY",linea,0);
       while (fgets ( linea, sizeof linea, stdin ) != NULL) {
              linea[strlen(linea)-1]=0;
              strcpy(L2,(char*)timestamp(linea,linea));
              l4l_printf(0xFFFFFFFF,L4L_MSJ_Comentar,L2);
       }
       l4l_end();
       return(0);
}
```

este código leerá de *stdin* todas las líneas y procederá, para cada una de ellas, a su registro no sin calcular previamente su timestamp y firma de cada linas.

Para la configuración timestamp, deberá crearse el fichero *timestamp.h* para la compilación de la librería. No son necesarias la información correspondiente al certificado digital ya que el timestamp es obtenido a via http, por lo que la fuerza de la seguridad de los hash encadenados será la contraseña indicada en línea de comandos (`/home/root/pass`):

```
#define TIMESTAMP_URL "http://www.originstamp.org/api/stamps"
#define TIMESTAMP_TOKEN "854081b6dad0c28adcd2f1907f012e3f"
#define TIMESTAMP_HEADER(T) "Authorization: Token token=%s",T
#define TIMESTAMP_POST(H,T) "{\"hash_sha256\" : \"%s\", \"title\":\"%s\"}",H,T
#define TIMESTAMP_HEADER_CONTENT "Content-Type: application/json"

//#define TIMESTAMP_SSLCERTTYPE "PEM"
//#define TIMESTAMP_SSLCERT "testkey.pem"
//#define TIMESTAMP_KEYPASSWD ""
//#define TIMESTAMP_SSLKEYTYPE "PEM"
//#define TIMESTAMP_SSLKEY "test"
//#define TIMESTAMP_CAINFO "cacert.pem"
//#define TIMESTAMP_SSL_VERIFYPEER 1L
```

Los ficheros y modificaciones anteriormente descritos se compilarán en el binario test.exe mediante el siguiente fichero makefile:

```
TARGET=test.exe

test.exe: libl4l.a test.c compcfg.h
gcc test.c -o test.exe -L. -ll4l -lcrypto  -lpthread -lrt -lcurl -std=gnu99

libl4l.a: l4l.o timestamp.o  log.cfg
ar -cvq libl4l.a l4l.o timestamp.o

l4l.o timestamp.o: l4l.c timestamp.c
gcc -c l4l.c timestamp.c

clean:
rm -fr test.exe libl4l.a l4l.o timestamp.o
```

Una vez obtenido el fichero test.exe podrá definirse el valor de la variable PROMPT_COMMAND para forzar el envío de información a registrar a una cola fifo:

```
PROMPT_COMMAND='printf " result=%s %s time=\"%s\" history>`history 1`"  $? `id|sed
"s/(.*//g"` `date "+%s.%N"` > /etc/l4l.fifo'
```

La definición de esta variable podrá incluirse con permisos de solo lectura dentro del fichero .bashrc de cada usuario. Y se ejecutará con el usuario root el siguiente comando para levantar el proceso que leerá todos los comandos ejecutados por todos los usuarios que hayan incluido esta definición en su arranque:

```
root> nohup ./test.exe test.pass < /etc/l4l.fifo
```

De no existir esta cola fifo deberá de haberse creado con anterioridad mediante el comando:

```
root> mkfifo /etc/l4l.fifo
```

De esta forma se generará en el fichero de registro `/var/log/.bash_history_<ficha>_<pid>.log` los comandos ejecutados, indicando el usuario y hora de sistema en el que se ejecutó (a parte del timestamp):

```
MSJ-Version: L4L v.0.3 Copyright (c) 2015 Jose Antonio Ortiz <jaortizb@gmail.com>
MSJ-License: This software is released under the LGPL 2.1 license, see the LICENSE file.
MSJ-Log: Log File Name /home/Carmen/.bash_history_20150822_172033_4876.log
MSJ-Bin: Binary file >./test.exe<
MSJ-Bin: Binary file arg[1]=test.pass
MSJ-Def: L4L_ROTATION: YES
MSJ-Def: L4L_HASH_METHOD: sha1
MSJ-Cfg: print_mask=0xffffffff, hash_mask=0xffffffff, sync_mask=0x0,flush_mask=0x0
MSJ-Cfg: buffsize=4098, bufftype=_IOFBF, max_size=5000000, max_seconds=3600
>dc5ac65843< MSJ-Comentar:
{"hash_sha256":"4c242f4d6ca31ee94be736ac996b3754b0ea50ef132f45dfe270700427ff4c89","created
_at":"2015-08-22T14:59:47.191Z","updated_at":"2015-08-
22T14:59:47.191Z","submitted_at":null,"title":"result=0 uid=1000 time=1440432527.594111400
history>  554  ls -ltr"}
>2d1c1d5987< MSJ-Comentar:
{"hash_sha256":"a31ee94be73ea50ef132f46ac996b3754b05dfe4c242f4d6c270427ff4c89070","created
_at":"2015-08-22T14:59:53.061Z","updated_at":"2015-08-
22T14:59:53.061Z","submitted_at":null,"title":"result=0 uid=1000 time=1440432533.841201310
history>  555  cd"}
```

Este es el registro correspondiente al usuario con uid igual a 1000 en el momento indicado y para los comandos *ls -lt*r y *cd.*

En la funcionalidad implementada faltarían por depurar importantes consideraciones de seguridad que garanticen que únicamente acceden a la cola fifo las sesiones bash que, de forma lícita, registren la ejecución de un comando. Podría asegurarse que la variable de entorno PROMPT_COMMAND sea invariablemente de solo lectura y que el acceso a la cola fifo solamente se acceda desde este comando definido en la variable PROMPT_COMMAND, mediante alguna configuración algo mas imaginativa de los *sudores* del sistema. O también, la forma mas eficaz de proceder, podría ser , la manipulación del fuente bash y que de su recompilación se asegure que actua como un *keylogger* que envíe las líneas de registro a un fichero de logs por usuario a una ruta protegida que impida la manipulación por un usuario que no sea root o tenga privilegios de la explotación del sistema o de su seguridad. Estas modificaciones en *bash* pueden intuirse rápidamente de la lectura del fuente *test.c*.

En cualquier caso el registro actual de comandos por parte del usuario, en su sesión linux/unix puede ponerse en entredicho en la mayor parte de las ocasiones y por lo tanto invalidar un informe forense.