2007

Forensic Log Investigator (FLI): a log analysis and visualization tool

Thieu Van Tran Pham
Iowa State University

Forensic Log Investigator (FLI) – a log analysis and visualization tool

by

Thieu Van Tran Pham

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Co-majors: Information Assurance; Computer Engineering

Program of Study Committee:
Yong Guan, Major Professor
Ying Cai
Steve Russell

Iowa State University

Ames, Iowa

2007

DEDICATION

To Dad and Mom – through the years, you have always been there for me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In a cyber crime investigation, investigators often have to examine and analyze log files, which contain valuable information – a history of actions, to reconstruct a chain of past events and ascertain whether or not a crime has been committed and the circumstances surrounding the crime.

There exist many types of log files such as server logs, firewall logs, intrusion-detection logs, system logs, application logs, and phone logs. Due to the lack of standard format, they follow their own arbitrary formats which present a formidable challenge and complexity to analysis. Additionally, with these log files containing a huge number of log entries, it is difficult, overwhelming, and daunting to extract the relevant evidence, analyze, keep track, and make sense of the information efficiently and reliably. Since each log contains only a little information, a fragment of the whole, it is particularly beneficial for investigators to examine logs together. Visualization allows the investigator to correlate the information, see the patterns, and gain insight into the events under examination.

This thesis provides the detailed design and implementation of FLI (Forensic Log Investigator). FLI is a powerful, advanced analysis and visualization tool built upon an enterprise infrastructure and the latest technologies to help computer forensics practitioners carry out investigations and perform analysis efficiently and effectively. FLI will do all the heavy work of processing, sorting, and searching of all the information and present the information to investigators in visual ways that investigators can easily understand, analyze, and extract relevant evidential information. Having FLI doing the heavy and tedious work allows investigators to devote more time to analysis and thereby results in solving the actual crime in a shorter time with fewer resources.

CHAPTER 1. INTRODUCTION

1.1 Background and Motivation

1.1.1 Overview of Computer Forensics and Computer Logs

As the world becomes more interconnected through the Internet and more government agencies and businesses moving their assets to the Internet, cyber crimes are getting more sophisticated, more organized, and on the rise. For governments, cyber crimes threaten national security. For businesses, cyber crimes damage profit and shake the trust of consumers. Unlike traditional crimes, cyber crimes operate in a different domain space – posing new and challenging problems. To better understand, prevent, and combat cyber crimes, Digital Forensic Science is born and defined by the Digital Forensic Research Workshop [1] as "The use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations." To determine whether or not a crime has been committed, a computer forensics investigation is carried out, often after the fact, to collect the relevant evidence which lead to detection and discovery of vulnerabilities and criminal activities. In turn, these findings lead to the prevention of exploitation and prosecution and conviction of criminals.

Most, if not all, computer programs produce log files. Log files are computer files that are produced by computer programs to record significant events as they happen while the programs are running. They provide important clues to knowing what happens at the time of an event which helps us understand how and why something happens. These log files provide a window into the internal working of a program at runtime helping programmers in

troubleshooting and fixing errors that exist in the program. Typically, they are text-based and specific to a program.

1.1.2 Digital Forensic Research

Digital Forensic Science is a relatively new field. In 2001, computer researchers, computer forensic examiners, and analysts came together at the First Digital Forensic Research Workshop (DFRWS) to exchange ideas and lay out a road map for digital forensic research. As figure 1.1 shows, the three different areas were identified. Depending on the area, the objectives and environment differ which influence the requirements of research.

Area	Primary Objective	Secondary Objective	Environment
Law Enforcement	Prosecution		After the fact
Military IW Operations	Continuity of Operations	Prosecution	Real Time
Business & Industry	Availability of Service	Prosecution	Real Time

Figure 1.1 Guidelines for Digital Forensic Research [1]

Similar to other fields, Digital Forensic Science has many challenges. Dr. Eugene Spafford has identified the following "Big Computer Forensic Challenges [1]:"

- **Technical**: "Keeping up" is a major dilemma. Digital technology continues to change rapidly. Terabyte disks and decreasing time to market are but two symptoms that cause investigators difficulty in applying currently available analytical tools. Add to this the unknown trust level of tools in development and the lack of experience and training so prevalent today and the major problems become very clear.
- **Procedural**: Currently, digital forensic analysts must collect everything which in the digital world leads to examination and scrutiny of volumes of data heretofore unheard of in support of investigations. Analytical procedures and protocols are not standardized nor do practitioners and researchers use standard terminology.

- **Social**: Individual privacy and the collection and analysis needs of investigators continue to collide. Uncertainty about the accuracy and efficacy of today's techniques causes data to be saved for very long time periods, which utilizes resources that may be applied toward real problem solving rather than storage.
- **Legal**: We can create the most advanced technology possible, but if it doesn't comply with the law it is moot.

The workshop also defined the process (shown in figure 1.2) used in digital forensic analysis which enables "practitioners to visualize where they need to add capability from what is available. Likewise, academic researchers will use the process to look for shortfalls in technology, helping them to focus on areas where research is needed the most [1]."

Identification	Preservation	Collection	Examination	Analysis	Presentation	Decision
Event/Crime Detection	Case Management	Preservation	Preservation	Preservation	Documentation	
Resolve Signture	Imaging Technologies	Approved Methods	Traceability	Traceability	Expert Testimony	
Profile Detection	Chain of Custody	Approved Software	Validation Techniques	Statistical	Clarification	
Anomalous Detection	Time Synch.	Approved Hardware	Filtering Techniques	Protocols	Mission Impact Statement	
Complaints		Legal Authority	Pattern Matching	Data Mining	Recommended Countermeasure	
System Monitoring		Lossless Compression	Hidden Data Discovery	Timeline	Statistical Interpretation	
Audit Analysis		Sampling	Hidden Data Extraction	Link		
Etc.		Data Reduction		Spacial		
		Recovery Techniques				

Figure 1.2 Investigative Process for Digital Forensic Science [1]

Each category has its challenges and requires more research. As you will see, the research of this thesis falls under several categories defined in this process, namely, collection, examination, analysis, and presentation.

For analysis, it can be broken into six categories [2]:

- **Physical Media Analysis:** The analysis of the physical media layer of abstraction, which translates a custom storage layout and contents to a standard interface, IDE or SCSI for example. The boundary layer is the bytes of the media. Examples include a hard disk, compact flash, and memory chips. The analysis of this layer includes processing the custom layout and even recovering deleted data after it has been overwritten, for example.
- **Media Management Analysis**: The analysis of the media management layer of abstraction, which organizes storage media. The boundary layer is another collection of bytes from the media. Examples of this layer include dividing a hard disk into partitions, organizing multiple disks into a volume, and integrating multiple memory chips into memory space. This layer may not exist in all types of media, for example a database may access an entire hard disk and not create partitions.
- **File System Analysis:** The analysis of the file system layer of abstraction, which translates the bytes and sectors of the partition to directories and files. The boundary layer is file content. The analysis in this layer includes viewing directory and file contents and recovering deleted files.
- **Application Analysis**: The analysis of the application layer of abstraction, which translates data, typically returned from the file system, into the custom format needed by the application. Analysis in this layer includes viewing log files, configuration files, images, documents and reverse engineering executables. The input data will typically come from the file system, but applications such as databases may read directly from the disk.

- **Network Analysis**: The analysis of the network layer of abstraction, which translates the lowest level data from a physical network or wireless network to the data that is used by an application. Analysis in this layer includes analyzing network packets and IDS alerts. Analysis of logs generated by network services, a firewall or web server for example, falls under Application Analysis.
- **Memory Analysis**: The analysis of the memory layer of abstraction, which translates the bytes of the memory media to processes and system data. Analysis in this area includes identifying the code that a process was running and extracting sensitive data that was not stored elsewhere.

Of the categories listed above, the research of this thesis falls under application and network analysis and helps in addressing the two big problems that exist in examination and analysis which are "the Complexity Problem" – "acquired data are typically at the lowest and most raw format, which is often too difficult for humans to understand. It is not necessarily impossible, but often the skill required to do so is great, and it is not efficient to require every forensic analyst to be able to do so. [2]" and "the Quantity Problem" – "the amount of data to analyze can be very large. It is inefficient to analyze every single piece of it. Data reduction techniques are used to solve this, by grouping data into one larger event or by removing known data. [2]"

Additionally, the workshop identified several major issues in this new discipline [1]:

- **Performance**: As research begins to accommodate analysis needed in the civilian and military sectors, speed will be a crucial factor in determining effectiveness and overall success. Of course, performance and speed are almost synonymous terms in computing. The major item affecting overall performance is data volume: the amount of data collected for analysis of this type is very often quite large. Historically, they have been very good collectors of bulk data but less than efficient in filtering and intelligent data reduction that will be necessary in network forensic analysis.

Therefore, research activities must focus in part on increasing the efficiency of those techniques designed to scrutinize reams of data and cull out whatever falls outside of investigative requirements.

- **Complexity:** The move from careful analysis of already complex standalone computers to scrutiny of data in networked environments significantly raises the level of detail, especially for the following investigative aids:
    - **Tools** designed for use in singular environments cannot easily transition to networks hosting multiple operating systems, protocols, applications, and data formats. Research is needed in providing tools and techniques that accommodate this mixture.
    - **Correlation** of the volumes of data into usable, understandable chunks or sections is necessary to afford useful analysis. Work in areas that help analysts and examiners understand data relationships in the volumes of information stored in intrusion detection databases is crucial.
- **Collaboration:** Due to the wider array of expertise required for collaboration analysis, it becomes even more of a concern that data regarding tools and techniques to aid analysis will be shared among all responsible parties in a timely and accurate fashion.

In Digital Forensic Science, as recognized and stated above, there are still many issues and research challenges that exist in all areas; and much research is needed to progress the field. This thesis addresses some of the issues and challenges aforementioned and contributes to the efforts of advancing the field of Digital Forensic Science.

1.1.3 Motivation

Edmund Locard's principle of exchange states that "wherever he steps, whatever he touches, whatever he leaves, even unconsciously, will serve as a silent witness against him."

Every action carried out always leaves behind a trace of its presence. Once consolidated and analyzed, these traces construct a chain of events – providing an understanding of what, where, when, how, and why those actions have happened. In a criminal investigation, these traces allow law enforcement to identify, capture, prosecute and convict the perpetrators. As one leaves physical traces of oneself in the physical world, one leaves digital traces in the computer world, often, in what so called "log files."

To better investigate, understand, and prevent a crime, these log entries can be correlated and studied. There are various types of logs such as server logs, firewall logs, intrusion-detection logs, system logs, and application logs, with each recording an action in the chain of actions. Even though they, in commonality, record some actions, they record different actions resulting in different outputs, each following its own format which makes it difficult to analyze. To untrained investigators, logs are chaotic. To trained investigators, logs are difficult to work with and overwhelming. Having the many different formats and contents also poses a real technical challenge, because they cannot simply be stored in a relational table. Relational databases work great with structured data. Structured data is data that can easily be divided into tables and columns. A log file by itself is structured data, but when dealing with a number of different logs together, they are not. As a result, to overcome this challenge and accommodate the many log formats and contents, a new technique has to be created.

These log files are usually large, containing lots of data; hence, it is extremely difficult to manually go through, find the relevant information, analyze, and keep track of the information. Whereas humans cannot process such amount of data efficiently and accurately, computers can do so more efficiently, reliably, and accurately. As already mentioned, it is a difficult and daunting task to extract the relevant evidence. Wouldn't it be nice to let the computer do all the heavy work of processing, sorting, and collecting all the relevant

information and present the information to the investigator in such a way that the investigator can easily understand and analyze.

Presentation of evidence is also a very important part of the investigative process. To get the court to convict an offender, evidence of the crime must be presented to the court. Often, the court does not understand technologies. We have always known that "a picture is worth a thousand words," so it is likely more effective to show the relevant evidence in a manner that the court can understand, instead of showing the printout of illegible text.

The Internet has revolutionized how computer-related work is done. No longer is a user constrained to one computer at one location. This has many benefits compared to the traditional model. FLI is created as a web application to leverage these benefits. An investigator can be anywhere in the world and still can work on his or her case using a web browser over the Internet. Also, multiple investigators can collaborate and share information on an investigation.

These are the motivations that have led to the creation of FLI (Forensic Log Investigator). FLI will ease the job of the investigator and help answer the questions:

- What crime has been committed?
- Who is involved or not involved in a crime?
- When did the crime occur?
- Where did the crime take place?
- How has the crime been committed?

The objective of this research is to design and implement FLI. To achieve this objective, many non-technical and technical challenges have had to be overcome. The rewarding result is that a useful tool has been created.

As a summary, here are the motivations:

- Reduce time, money, and effort spent in an investigation
- Work with the different log formats and contents
- Handle and manage large amount of log data efficiently
- Visualize data to help better understand, discover interesting patterns, and explain findings
- Provide accessibility and availability to users regardless of physical locations
- Centralize access for performance, maintenance, security, scalability, and upgradability
- Correlate logs to help investigators understand data relationships in the volumes of information stored
- Facilitate collaboration and sharing of ideas and expertise among investigators in an investigation
- Keep track of findings and evidence in a case
- Associate computers to physical locations to further the investigation
- Have satisfying experience for users through intuitive, interactive, rich, and user-friendly interface
- Search, filter, sort, and organize data for quick and easy retrieval

1.2 Overview of FLI

FLI is a powerful, cross-platform, multi-user web application that aims to significantly ease the job of an investigator. In a computer forensics investigation, FLI helps in examination, analysis, and presentation of data. With FLI, the investigator can analyze and interpret the data more efficiently and accurately.

FLI has various components that work together to fulfill the overall objective.

- **FLI Log Parsing Framework:** The investigator can work with different types of log files using the different adapters.
- **FLI Timeline:** The investigator can graphically and easily go back and forth through the timeline and see the sequence of events that have happened.
- **FLI Filter & Query**: The investigator can selectively filter out and inspect only those events that the investigator is interested in and not be overwhelmed with the vast amount of data available.
- **FLI Charts:** The investigator can group, view, and present the information visually through charts.
- **FLI Trace:** The investigator can trace the events through time or the path taken by a web user in a graphical manner.
- **FLI Viewer:** The investigator can search the raw log data.
- **FLI Grid:** The investigator can find and sort the information quickly and easily.
- **FLI Map:** The investigator can easily locate and pick out the country that a user resides.
- **FLI Note & Notebook:** The investigator can take notes and bookmark things that require further investigation or can be assembled into a report.

CHAPTER 2. DESIGN AND ARCHITECTURE

2.1 Overview

FLI is architected to be high-performance, cross-platform, and highly available web application, comprising of a server component and a client component.

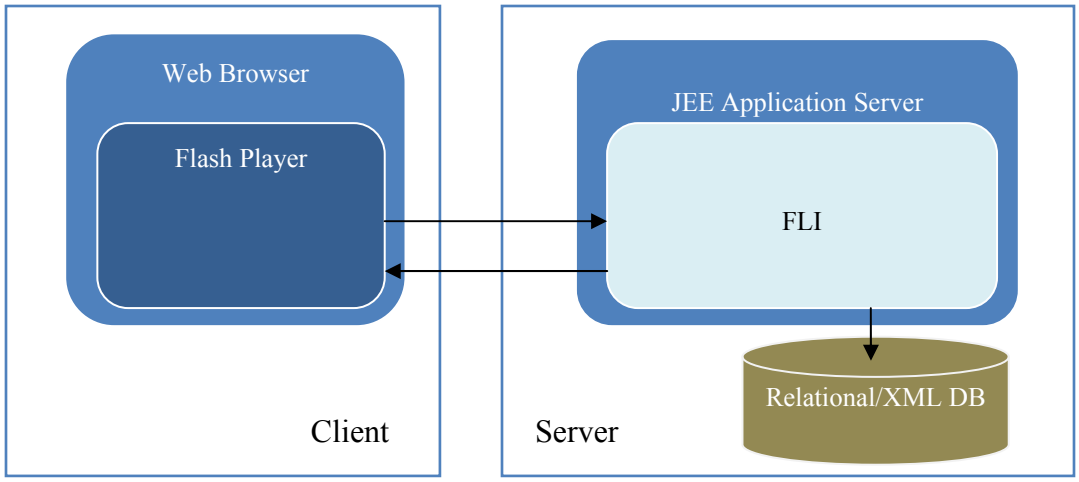

Figure 2.1 FLI Overall Architecture

The server component runs in a Java Virtual Machine which means that it is independent of the underlying platform whether it is Unix, Linux, or Windows. The server provides performance and scalability – as demand increases, the server can be clustered with other servers to meet the demand. Since it runs in one central place, it is easy to maintain, upgrade, and audit users' activities.

The client component runs in Adobe Flash Player, which extends the functionalities of the web browser and is readily available in most browsers, providing users with a rich and satisfying experience.

2.2 FLI Log Parsing Framework

For the tool to be able to work with and for investigators to make sense of logs, having a variety of formats and contents, they have to be transformed into a common format. This is the objective of the FLI Log Parsing Framework. The framework provides flexible and extensible facilities to convert different log formats into the FLIEvent format (defined in the next section).

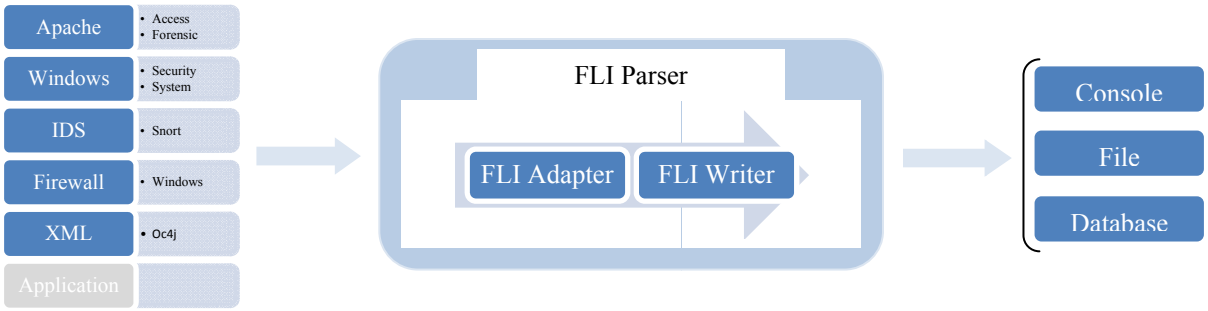

Figure 2.2 FLI Log Parsing Framework

Through FLI Log Parsing Framework, a log file is processed, transformed, and written out to a data source – a console, a file or a database. The process is explained in the following flowchart.

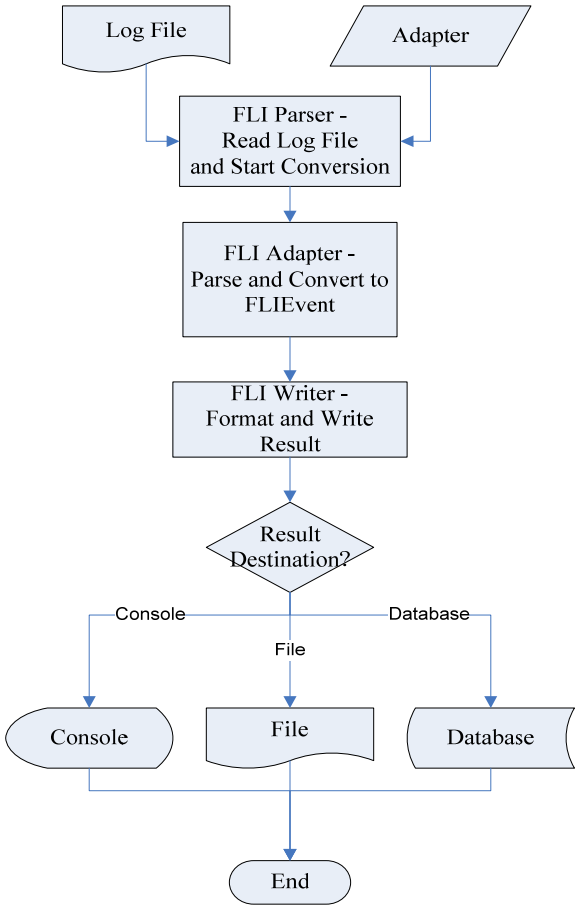

Figure 2.3 FLI Log Parsing Flowchart

2.2.1 FLI Log Format – FLIEvent Schema

FLIEvent format is designed to provide flexibility and extensibility to accommodate the numerous, different log formats and contents. Furthermore, it is designed to work efficiently with FLI. It is XML-based which reaps the benefits of XML such as portability, human readability, interoperability, etc. It is defined as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.thieupham.com/FLIEvent/1.0"
            xmlns="http://www.thieupham.com/FLIEvent/1.0"
            elementFormDefault="qualified">
      <xs:annotation>
            <xs:documentation>
                  FLI (Forensic Log Investigator) Event Schema
                  Author:  Thieu Pham
            </xs:documentation>
      </xs:annotation>
      <xs:element name="FLIEventSet">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element
                              name="FLIEvent" type="FLIEventType"
                              minOccurs="1" maxOccurs="unbounded"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
      <xs:element name="FLIEvent" type="FLIEventType"/>
      <xs:complexType name="FLIEventType">
            <xs:sequence>
                  <xs:any minOccurs="0" maxOccurs="unbounded"/>
                  <xs:element
                        name="Extension" type="ExtensionType"
                        minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="type" type="xs:string" use="required"/>
            <xs:attribute
                        name="occurredTime" type="xs:dateTime"
                        use="required"/>
            <xs:attribute
                        name="description" type="xs:string"
                        use="required"/>
            <xs:attribute name="key" type="xs:string" use="optional"/>
      </xs:complexType>
      <xs:complexType name="ExtensionType">
            <xs:sequence>
                  <xs:any minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="type" type="xs:string" use="required"/>
            <xs:attribute
                        name="description" type="xs:string"
                        use="required"/>
      </xs:complexType>
</xs:schema>
```

- FLIEventSet – contains a set of FLIEvent elements
- FLIEvent – represents an event that has been logged
    - Attributes:
        - Type – type of the log
        - OccurredTime – the time that the event occurs in the format yyyy-MM-ddTHH:mm:ss
        - Description – a short description of the event or a summary
        - Key – this is used for merging different FLIEvent elements into one FLIEvent element
    - Elements:
        - Can contain any number of valid, user-defined elements
        - Extension – can use for any purpose but is usually used to group related data into a meaning element

Section 2.2.3 will show the details of several types of logs being converted into FLIEvent format to help clarify FLIEvent.

2.2.2 FLI Adapters

FLI provides different adapters to interface with the different log formats. An adapter's job is to convert a log format into FLIEvent. The adapters fall into two categories: generic adapter and concrete adapter. A generic adapter works with category of logs. A concrete adapter is an instance of a generic adapter supplying its own parameters to work specifically with one type of log.

2.2.2.1 FLI XML Adapter

This adapter works with XML-based logs.

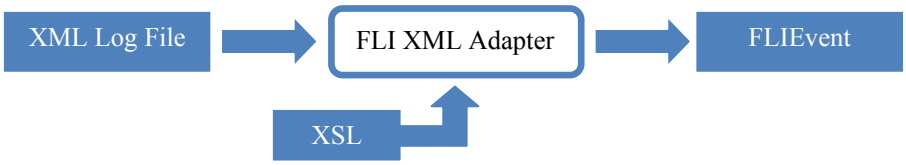

Figure 2.4 FLI XML Adapter

In addition to the log file, the adapter takes a XSL file. Here's the XSL file for OC4J logs:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output omit-xml-declaration="yes"/>
   <xsl:template match="/">
            <xsl:element name="FLIEvent">
                  <xsl:attribute name="type">
                        <xsl:value-of
                        select="/MESSAGE/HEADER/COMPONENT_ID"/>
                  </xsl:attribute>
                  <xsl:attribute name="occurredTime">
                        <xsl:value-of
                              select="/MESSAGE/HEADER/TSTZ_ORIGINATING"/>
                  </xsl:attribute>
                  <xsl:attribute name="description">
                        <xsl:value-of select="/MESSAGE/PAYLOAD/MSG_TEXT"/>
                  </xsl:attribute>
                  <xsl:attribute name="severity">
                        <xsl:value-of select="/MESSAGE/HEADER/MSG_LEVEL"/>
                  </xsl:attribute>
```

```
                    <xsl:element name="MessageType">
                          <xsl:value-of
                                select="/MESSAGE/HEADER/MSG_TYPE/@TYPE"/>
                    </xsl:element>
                    <xsl:element name="Host">
                          <xsl:value-of select="/MESSAGE/HEADER/HOST_ID"/>
                    </xsl:element>
                    <xsl:element name="IP">
                          <xsl:value-of
                                select="/MESSAGE/HEADER/HOST_NWADDR"/>
                    </xsl:element>
                    <xsl:element name="User">
                          <xsl:value-of select="/MESSAGE/HEADER/USER_ID"/>
                    </xsl:element>
                    <xsl:element name="Module">
                          <xsl:value-of select="/MESSAGE/HEADER/MODULE_ID"/>
                    </xsl:element>
                    <xsl:element name="Thread">
                          <xsl:value-of select="/MESSAGE/HEADER/THREAD_ID"/>
                    </xsl:element>
              </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

2.2.2.2 FLI Regular Expression Adapter

This adapter works with text-based logs with formats such that its various parts can be expressed by a regular expression.

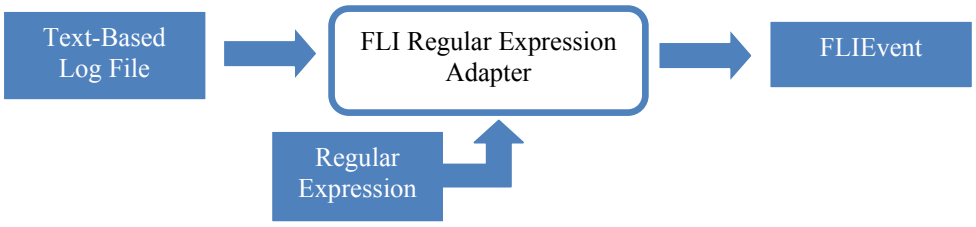

Figure 2.5 FLI Regular Expression Adapter

Here are two examples of the regular expressions used for Apache HTTP Server access logs and Windows firewall logs:

Apache HTTP Server Access:

```
219.95.235.26 - - [31/Mar/2007:08:10:39 -0500] "GET /index.html HTTP/1.1"
302   297   "http://search.yahoo.com/search?&p=timkiemban+.com&fr=ieas-dns"
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)"
```

Regular Expression:

```
^([\d.]+) (\S+) (\S+) [([\w:/]+\s[+\-]\d{4})\] "(.+?)" (\d{3}) (\d{3})
"(.+?)?" "(.+?)"
```

Windows Firewall:

```
2007-10-11 20:12:26 DROP TCP 72.14.203.99 192.168.0.4 80 1076 48 SA
885061199 3398995105 5720 - - - RECEIVE
```

Regular Expression:

```
^(\d{4}-\d{2}-
\d{2}\s\d{2}:\d{2}:\d{2})\s(.+)\s(.+)\s(.+)\s(.+)\s(.+)\s(.+)\s(.+)\
s(.+)\s(.+)\s(.+)\s(.+)\s(.+)\s(.+)\s(.+)
```

2.2.2.3 FLI PCAP Adapter

This adapter works with PCAP (application programming interface for packet capturing) dump file format produced by packet capturing applications such as tcpdump or Snort.

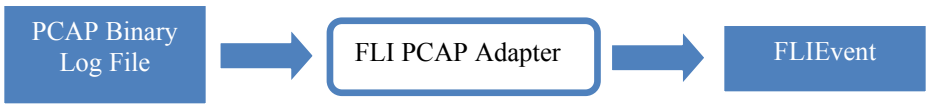

Figure 2.6 FLI PCAP Adapter

2.2.2.4 FLI Static Adapter

If the above adapters do not fulfill the need, the investigator can write his or her static class with FLI interfaces.

Figure 2.7 FLI Static Adapter

2.2.2.5 FLI GLA Adapter

FLI provides this adapter to work with GLA [22] adapters – taking advantage of those concrete adapters already written for GLA. To allow FLI to work with GLA adapters, FLI

extends GLA by writing its own outputter to return FLIEvent as the result and, furthermore, write the result to the database, something that GLA cannot do.

This adapter takes a GLA Adapter as an input and is the only one that utilizes the GLA API.

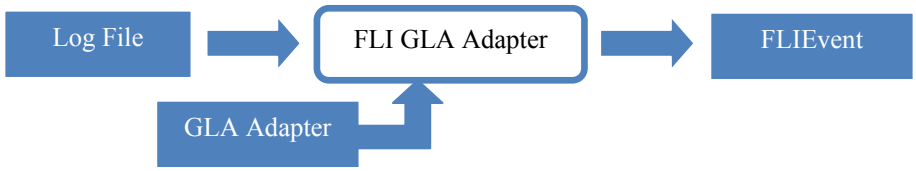

Figure 2.8 FLI GLA Adapter

2.2.3 Conversion of Different Log Formats to FLIEvent Format

Using FLI adapters, different log formats are parsed and converted into FLIEvent. The results of the transformations are shown below. Each of following log file is chosen to represent a category of logs and to demonstrate the power of the FLI adapters and the flexibility of the FLIEvent format to work with and accommodate the different log formats and contents. Also, FLI explores and recommends the use of Apache HTTP Server Forensic Log. Most people are not aware of this available feature, and it is a source of valuable information.

2.2.3.1 Apache HTTP Server Access Log

Apache HTTP Server records requests coming to the server in the well-known log file called the access log. Below is a request recorded in the Combined Log Format.

INPUT:

```
127.0.0.1 - - [04/Oct/2007:08:19:00 -0500] "GET / HTTP/1.1" 200 44 "-"
"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322;
.NET CLR 2.0.50727)" 1bc:4704e844:0
```

RESULT:

```
<FLIEvent
      type="Apache Access"
      occurredTime="2007-10-04T08:19:00"
      description="&quot;GET / HTTP/1.1&quot; 200 44 &quot;-&quot;
      &quot;Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR
      1.1.4322; .NET CLR 2.0.50727)&quot; 1bc:4704e844:0"
      severity="10"
      key="1bc:4704e844:0">
   <IP>127.0.0.1</IP>
   <ReturnCode>200</ReturnCode>
   <File>/</File>
   <Method>GET</Method>
   <Referer>-</Referer>
   <UserAgent>
      Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR
      1.1.4322; .NET CLR 2.0.50727) 1bc:4704e844:0
   </UserAgent>
<FileSize>44</FileSize>
</FLIEvent>
```

2.2.3.2 Apache HTTP Server Forensic Log

Unlike the Access Log, the Forensic Log is not well-known and seldom used. The Forensic Log is the output of the mod_log_forensic module which is available starting with version 1.3.30 of the Apache HTTP Server. The module extends the Access Log and records all HTTP request headers. These HTTP headers contain a lot of information that provide important and valuable clues in an investigation. It is highly recommended that administrators of web sites turn this feature on. By default, it is not loaded. To turn it on, make the following modifications in the httpd.conf.

- **Load the module.**

  ```
  LoadModule log_forensic_module modules/mod_log_forensic.so
  ```

  The mod_log_forensic module records all HTTP request headers in a separate log file with a unique identifier for each request. To correlate each request in the forensic log file with the request in the access log file, you have to modify the access log format to include `%{forensic-id}n` such as follows.

  ```
  LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-
  Agent}i\" %{forensic-id}n" ForensicCombined
  ```

- **Change your directives.**

```
CustomLog logs/access.log ForensicCombined
ForensicLog logs/forensic_log.log
```

Here's how it looks in the log files.

Access.log

```
127.0.0.1 - - [04/Oct/2007:08:19:00 -0500] "GET / HTTP/1.1" 200 44
"-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR
1.1.4322; .NET CLR 2.0.50727)" 1bc:4704e844:0
```

Forensic_log.log

```
+1bc:4704e844:0|GET / HTTP/1.1|Accept:image/gif, image/x-xbitmap,
image/jpeg, image/pjpeg, application/x-shockwave-flash,
application/vnd.ms-excel, application/vnd.ms-powerpoint,
application/msword, */*|Accept-Language:en-us|UA-CPU:x86|Accept-
Encoding:gzip, deflate|User-Agent:Mozilla/4.0 (compatible; MSIE 7.0;
Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR
2.0.50727)|Host:localhost|Connection:Keep-Alive
```

For analysis, we need to combine these two into one FLIEvent, because the headers do not make much sense being detached from the request. To accomplish this, the FLIEvent merger is used. This is the scenario where the key attribute provides this mean to merge different logs.

- **Convert access log entry into FLIEvent. Notice the key attribute.**

```
<FLIEvent
      type=" Apache Access"
      occurredTime="2007-10-04T08:19:00"
      description="&quot;GET / HTTP/1.1&quot; 200 44 &quot;-&quot;
      &quot;Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET
      CLR 1.1.4322; .NET CLR 2.0.50727)&quot; 1bc:4704e844:0"
      severity="10"
      key="1bc:4704e844:0">
  <IP>127.0.0.1</IP>
  <ReturnCode>200</ReturnCode>
  <File>/</File>
  <Method>GET</Method>
  <Referer>-</Referer>
  <UserAgent>
      Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR
      1.1.4322; .NET CLR 2.0.50727)
  </UserAgent>
  <FileSize>44</FileSize>
</FLIEvent>
```

- **Convert forensic log entry into FLIEvent.**

```
<FLIEvent
      type=" Apache Forensic"
      occurredTime=""
      description=""
      key="1bc:4704e844:0">
  <Accept>
      image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
      application/x-shockwave-flash, application/vnd.ms-excel,
      application/vnd.ms-powerpoint, application/msword, */*
  </Accept>
  <Accept-Language>en-us</Accept-Language>
  <UA-CPU>x86</UA-CPU>
  <Accept-Encoding>gzip, deflate</Accept-Encoding>
  <User-Agent>
      Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR
      1.1.4322; .NET CLR 2.0.50727)
</User-Agent>
  <Host>localhost</Host>
  <Connection>Keep-Alive</Connection>
</FLIEvent>
```

- **Merged result after running through FLIEventMerger.**

```
<FLIEvent
      type=" Apache Access And Forensic"
      occurredTime="2007-10-04T08:19:00"
      description="&quot;GET / HTTP/1.1&quot; 200 44 &quot;-&quot;
      &quot;Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET
      CLR 1.1.4322; .NET CLR 2.0.50727)&quot; 1bc:4704e844:0">
   <IP>127.0.0.1</IP>
   <ReturnCode>200</ReturnCode>
   <File>/</File>
   <Method>GET</Method>
   <Referer>-</Referer>
   <UserAgent>
      Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR
      1.1.4322; .NET CLR 2.0.50727) 1bc:4704e844:0
   </UserAgent>
   <FileSize>44</FileSize>
   <Extension type="HTTP Header" description="Apache Forensic">
      <Accept>
            image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
            application/x-shockwave-flash, application/vnd.ms-excel,
            application/vnd.ms-powerpoint, application/msword, */*
      </Accept>
      <Accept-Language>en-us</Accept-Language>
      <UA-CPU>x86</UA-CPU>
      <Accept-Encoding>gzip, deflate</Accept-Encoding>
      <User-Agent>
            Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET
            CLR 1.1.4322; .NET CLR 2.0.50727)
```

```
            </User-Agent>
            <Host>localhost</Host>
            <Connection>Keep-Alive</Connection>
         </Extension>
      </FLIEvent>
```

2.2.3.3 Snort Log

The log file [30] contains binary as its content with pcap dump format. Since, it is binary, its raw data cannot be included here in a readable format.

RESULT:

```
<FLIEvent type="Snort" occurredTime="2000-04-26T06:52:09.0"
      description="172.16.1.107 > 24.112.167.35 FTP Request=PASS
      temp123\r\n">
      <source>172.16.1.107</source>
      <destination>24.112.167.35</destination>
      <Extension type="Ethernet2"
            description="0:10:4b:70:14:e7 &gt; 0:10:4b:72:e5:37
            Protocol=0x800 (IP) Internet protocol (v4 or v6)">
            <dst>0:10:4b:72:e5:37</dst>
            <src>0:10:4b:70:14:e7</src>
            <proto>2048</proto>
            <crc>842206474</crc>
      </Extension>
      <Extension type="IPv4"
            description="172.16.1.107 &gt; 24.112.167.35 [ DF ] Protocol=6
            Id=0x4b62 TTL=64 Offset=0 Len=46">
            <ver>4</ver>
            <hlen>5</hlen>
            <tos>16</tos>
            <len>66</len>
            <id>19298</id>
            <flags>2</flags>
            <offset>0</offset>
            <ttl>64</ttl>
            <protocol>6</protocol>
            <checksum>33333</checksum>
            <saddr>172.16.1.107</saddr>
            <daddr>24.112.167.35</daddr>
      </Extension>
      <Extension type="TCP"
            description="1084 &gt; 21 [ ACK PSH ] Seq=1596650388
            Ack=2209684074 win=32120 Len=">
            <sport>1084</sport>
            <dport>21</dport>
            <seq>1596650388</seq>
            <ack>2209684074</ack>
            <hlen>8</hlen>
            <res>0</res>
            <code>24</code>
```

```
            <win>32120</win>
            <crc>63627</crc>
            <urg>0</urg>
            <option>1</option>
            <option>1</option>
            <option>8</option>
            <len>10</len>
            <tsval>144077926</tsval>
            <tsecr>8144903</tsecr>
      </Extension>
      <Extension type="FTP" description="Request=PASS temp123\r\n">
            <Request>PASS temp123\r\n</Request>
      </Extension>
</FLIEvent>
```

2.2.3.4 Windows Firewall Log

INPUT:

```
2007-10-18 10:23:43 DROP UDP 192.168.10.104 239.255.255.250 1900 1900 302
- - - - - - - - RECEIVE
```

RESULT:

```
<FLIEvent type="Windows Firewall" occurredTime="2007-10-18T10:23:43"
      description="DROP UDP 192.168.10.104 239.255.255.250 ">
      <action>DROP</action>
      <protocol>UDP</protocol>
      <src-ip>192.168.10.104</src-ip>
      <dst-ip>239.255.255.250</dst-ip>
      <src-port>1900</src-port>
      <dst-port>1900</dst-port>
      <size>302</size>
      <tcpflags>-</tcpflags>
      <tcpsyn>-</tcpsyn>
      <tcpack>-</tcpack>
      <tcpwin>-</tcpwin>
      <icmptype>-</icmptype>
      <icmpcode>-</icmpcode>
      <info>-</info>
      <path>RECEIVE</path>
</FLIEvent>
```

2.2.3.5 Windows System Log

RESULT:

```
<FLIEvent type="Windows System" occurredTime="2007-09-08T07:18:59"
      description="Microsoft (R) Windows (R) 5.01. 2600 Service Pack 2
      Multiprocessor Free."
      severity="20">
      <Username>N/A</Username>
```

```
      <EventCategory>0</EventCategory>
</FLIEvent>
```

2.2.3.6 Windows Security Log

RESULT:

```
<FLIEvent type="Windows" occurredTime="2007-09-10T09:49:09"
      description="Logon attempt by: MICROSOFT_AUTHENTICATION_PACKAGE_V1_0
      Logon account:  Administrator    Source Workstation: THIEU    Error
      Code: 0xC000006A"
      severity="16" key="1.0.1">
      <Username>SYSTEM</Username>
      <EventCategory>9</EventCategory>
</FLIEvent>
```

2.2.3.7 Oracle Containers for Java Log

INPUT:

```
<MESSAGE>
  <HEADER>
    <TSTZ_ORIGINATING>2007-07-20T09:29:20.059-05:00</TSTZ_ORIGINATING>
    <COMPONENT_ID>oc4j</COMPONENT_ID>
    <MSG_TYPE TYPE="NOTIFICATION"></MSG_TYPE>
    <MSG_LEVEL>1</MSG_LEVEL>
    <HOST_ID>TPCOMPUTER</HOST_ID>
    <HOST_NWADDR>192.168.10.29</HOST_NWADDR>
    <MODULE_ID>util.FileUtils</MODULE_ID>
    <THREAD_ID>10</THREAD_ID>
    <USER_ID>thieu</USER_ID>
  </HEADER>
  <CORRELATION_DATA>
      <EXEC_CONTEXT_ID><UNIQUE_ID>192.168.10.29:28398:1184941760075:0</UNI
      QUE_ID><SEQ>0</SEQ></EXEC_CONTEXT_ID>
  </CORRELATION_DATA>
  <PAYLOAD>
      <MSG_TEXT>
      Auto-unpacking C:\Oracle\OracleAS\j2ee\home\applications\dms.war...
      </MSG_TEXT>
  </PAYLOAD>
</MESSAGE>
```

RESULT:

```
<FLIEvent type="oc4j" occurredTime="2007-07-20T09:29:20"
      description="Auto-unpacking
      C:\Oracle\OracleAS\j2ee\home\applications\dms.war... "
      severity="1">
      <MessageType>NOTIFICATION</MessageType>
      <Host>TPCOMPUTER</Host>
```

```
      <IP>192.168.10.29</IP>
      <User>thieu</User>
      <Module>util.FileUtils</Module>
      <Thread>10</Thread>
</FLIEvent>
```

2.2.3.8 Cingular Log

INPUT:

```
9/22;01:30;PM; INCOMING;515-779-7738; NW;2
```

RESULT:

```
<FLIEvent type="CINGULAR" occurredTime="2007-09-22T01:30:00"
      description="INCOMING 515-778-7737" severity="" key="1.0.1">
      <CALL_TO>INCOMING</CALL_TO>
      <NUMBER_CALLED>515-778-7737</NUMBER_CALLED>
      <RATE>NW</RATE>
      <MINUTES>2</MINUTES>
</FLIEvent>
```

2.2.3.9 Other Logs

FLI Log Parsing Framework is powerful and flexible. With the generic adapters provided, other types of logs can be easily transformed into FLIEvent and incorporated into FLI.

2.3 FLI Modules

FLI is composed of a number of components that collaborate to deliver the solution. Figure 2.9 shows the various components, and each one will be explained in detail in the following subsections.

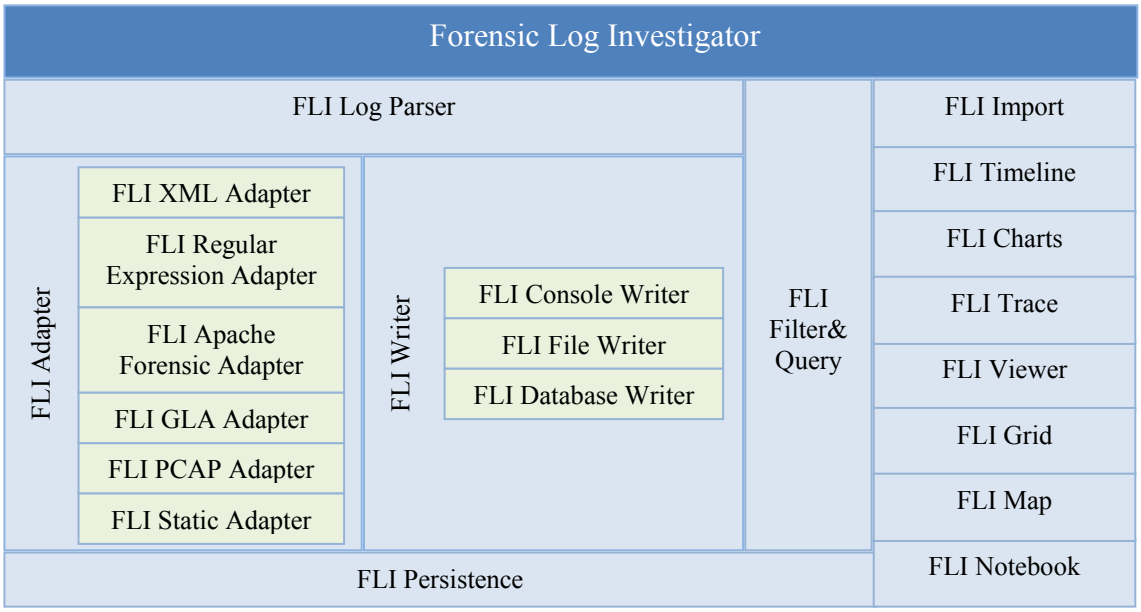

Figure 2.9 FLI Modules

2.3.1 FLI Filter & Query

Working with large volumes of data, the investigator needs the ability to filter the data, narrow down the data set, and search only for that relevant information the investigator wants to examine and analyze.

FLI Filter & Query provides a powerful facility to do this using SQL/XML as its querying language. SQL/XML is a mixture SQL and XPath. Most people are already familiar with SQL; however, not many people are familiar with XPath; therefore, quick reference to simple XPath expressions used in FLI is provided here. To write more complex expressions, you have to learn XPath which is beyond the scope of this document. More information regarding SQL/XML and XPath are provided in chapter 3.

Operator	Description
=	Equal
!=	Not equal
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
or	or
and	and

Table 2.1 XPath Operators

Filter By Attributes Using = Operator	
XPath Expression	**Description**
//@type="?"	Selects events with type
//@occurredTime="?"	Selects events with occurred time
//@description="?"	Selects events with description
//@severity="?"	Selects events with severity
Filter By Child Elements For Apache Access FLIEvent can any number of different child elements, so it is dependent on each type of log. The expressions written here are for the Apache access log.	
XPath Expression	**Description**
//IP="?"	Selects events with ip
//File="?"	Select events with request (web page path)
//Method="?"	Selects events with HTTP method (POST, GET, PUT, etc.)
Combination Using "and" and "or" operators	
XPath Expression	**Description**
//@type="?" and //@severity="?"	Selects events with type and severity
//@severity="?" and //IP="?"	Selects events with severity and ip
//IP="?" or //IP="?"	Select events for two specified IP addresses
Partial Matching	
ora:contains(/FLIEvent/@description,"user")>0	Partial search on a field

Table 2.2 FLI XPath Expressions

Here's sample expression using the some of functions defined above:

```
(//source="172.16.1.107" or //destination="172.16.1.107")
and ora:contains(/FLIEvent/@description,"telnet")>0
```

2.3.2 FLI Import

Before a log file can be processed by FLI, it has to be uploaded to the server and imported into FLI. Assuming that a log file has already been uploaded to the server, to import it into FLI, the user chooses the log file, the adapter, fill out the information about this log file. The process is defined in the following flowchart.

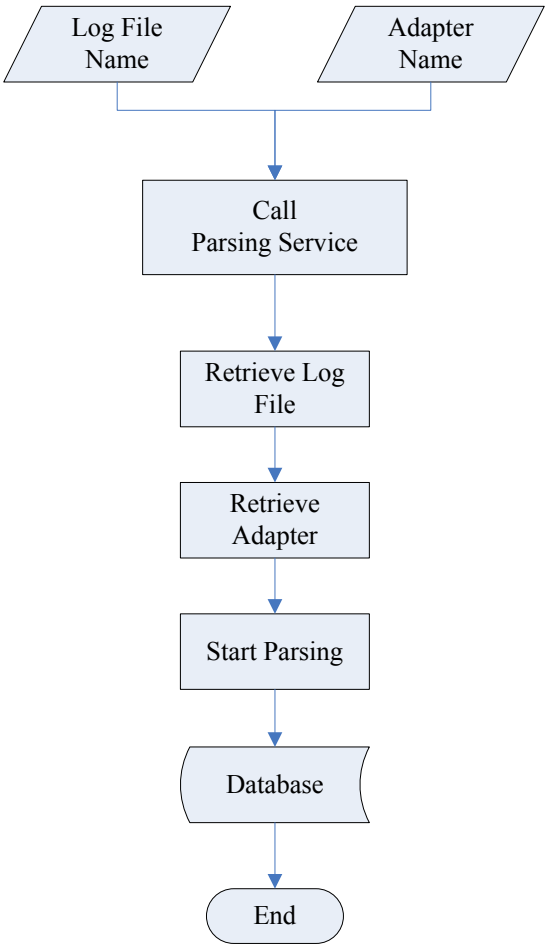

Figure 2.10 FLI Import Flowchart

When the parsing process is running, the lines turn red and the status window pops up. The user has the option to abort the process at anytime.

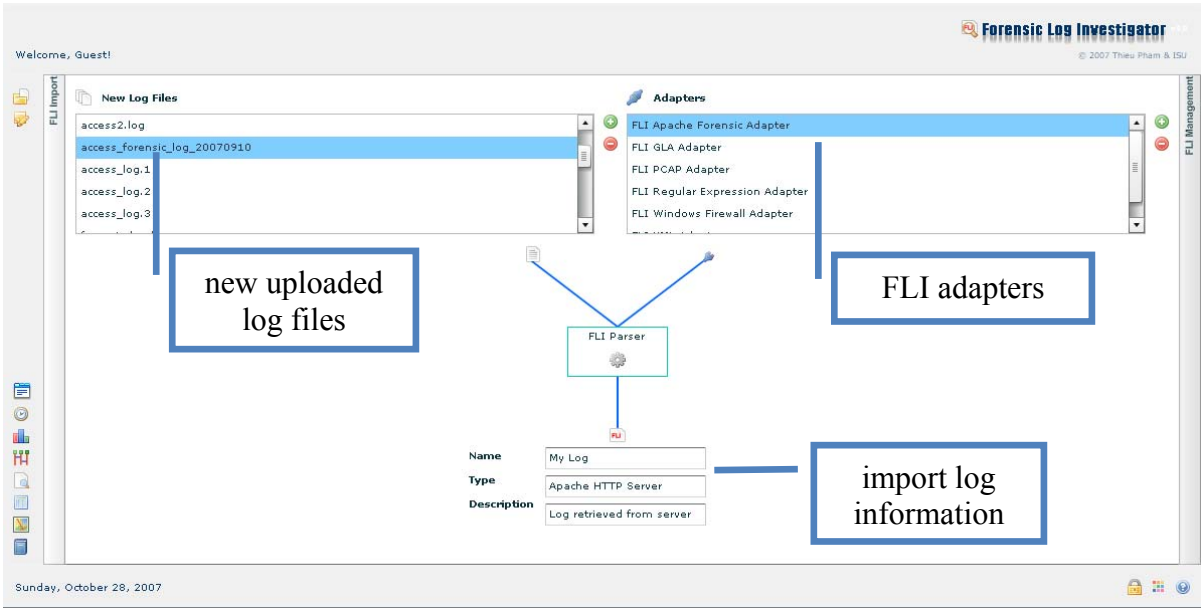

Figure 2.11 FLI Import - Initiation

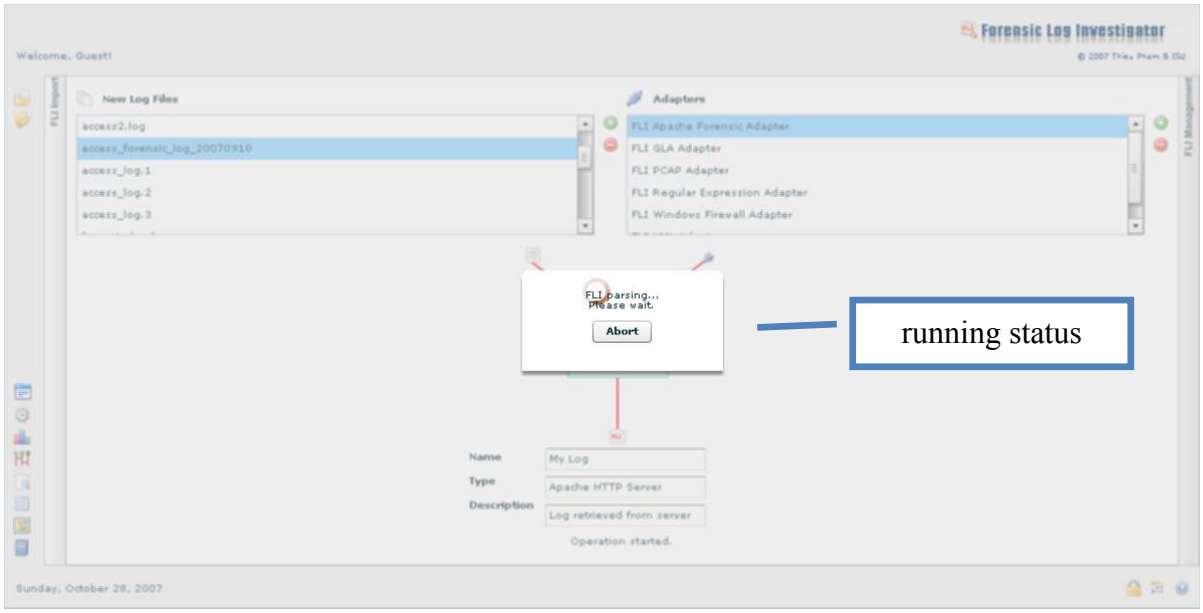

Figure 2.12 FLI Import – Running Status

2.3.3 FLI Timeline

FLI Timeline displays events on a timeline. "The timeline of computer events may provide a critical piece of information relating to the prosecution of involved persons. This information can help to pinpoint the location of certain individuals, can assist with the determination of alibis, can uncover conversations and correspondences, and can possibly help to ultimately determine the guilt or innocence of those facing criminal charges. [3]" FLI

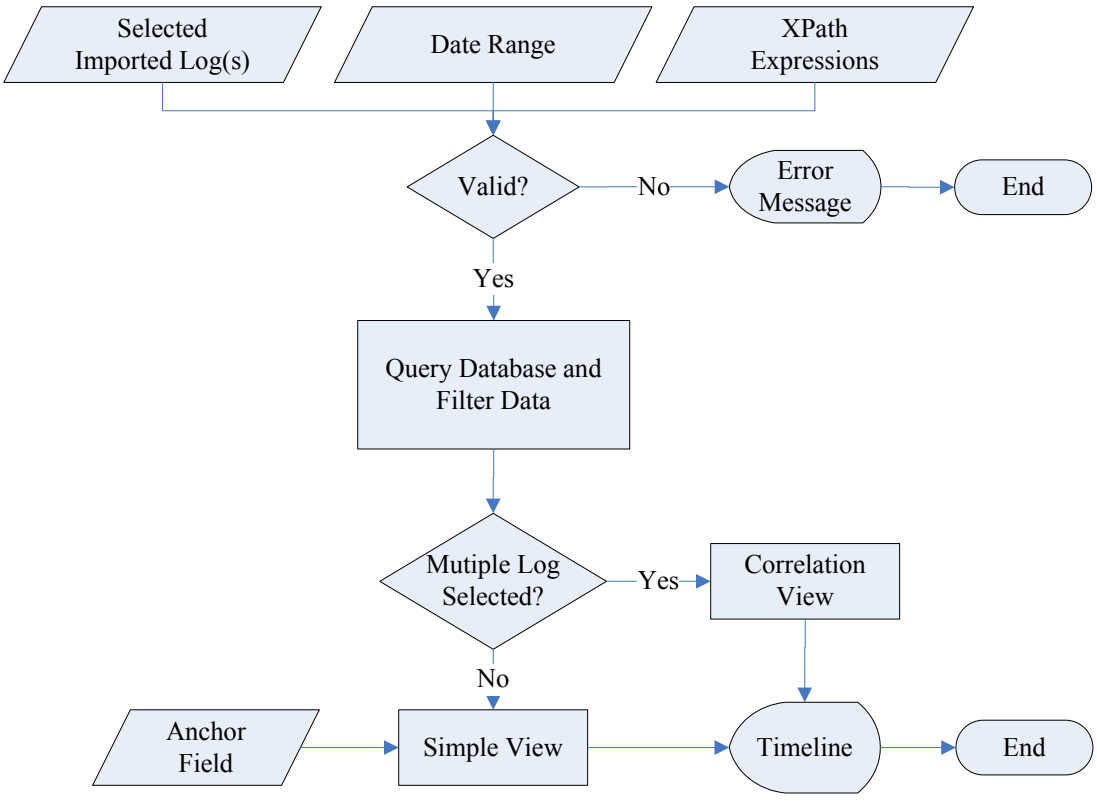

Figure 2.13 FLI Timeline – Select Simple/Multiple View Flowchart

Timeline provides the investigator a nice way to visually examine events either individually or collectively. The investigator can correlate different logs and view them together. FLI Timeline has two views: simple view (viewing only one log) and correlation view (viewing multiple, time-correlated logs). Figure 2.13 shows the flowchart of how a view is chosen.

The following figures (2.14, 2.15, 2.16, 2.18, 2.19, 2.20) show what can be done with FLI Timeline and see how this feature will help the investigator in an analysis.

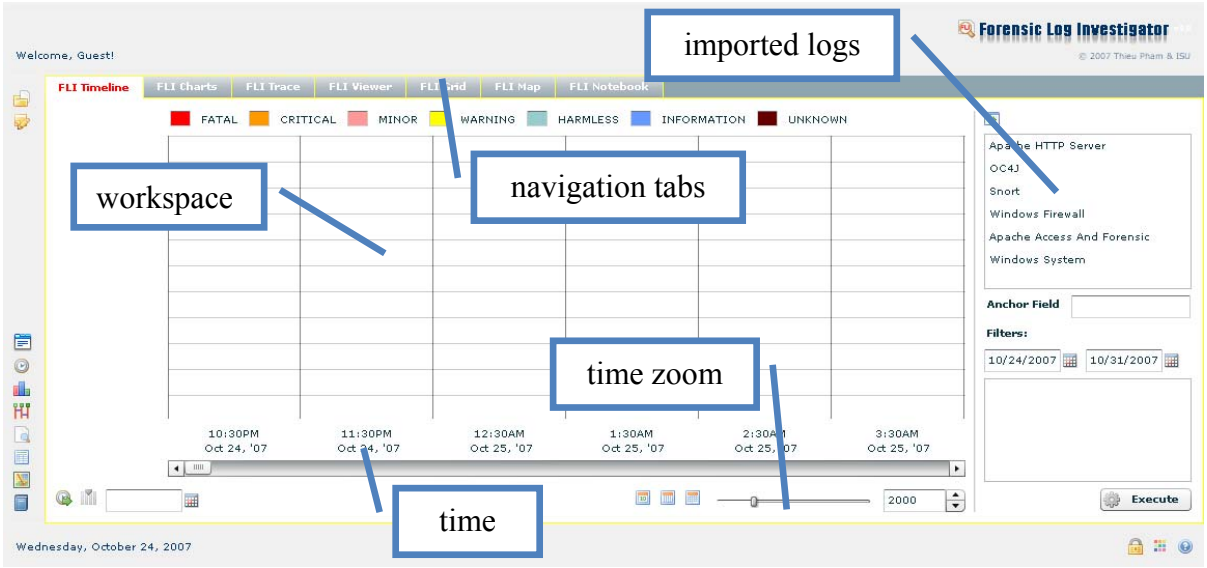

Figure 2.14 FLI Timeline – Layout

Here's the main layout. At the bottom, the date and time are shown. The investigator can scroll forward and backward through time using the scrollbar. With the time zoom, the investigator can scale the time precision which affects the items being shown in the workspace. This gives the investigator a quick way to walk through time and focus only on the events that the investigator is interested in.

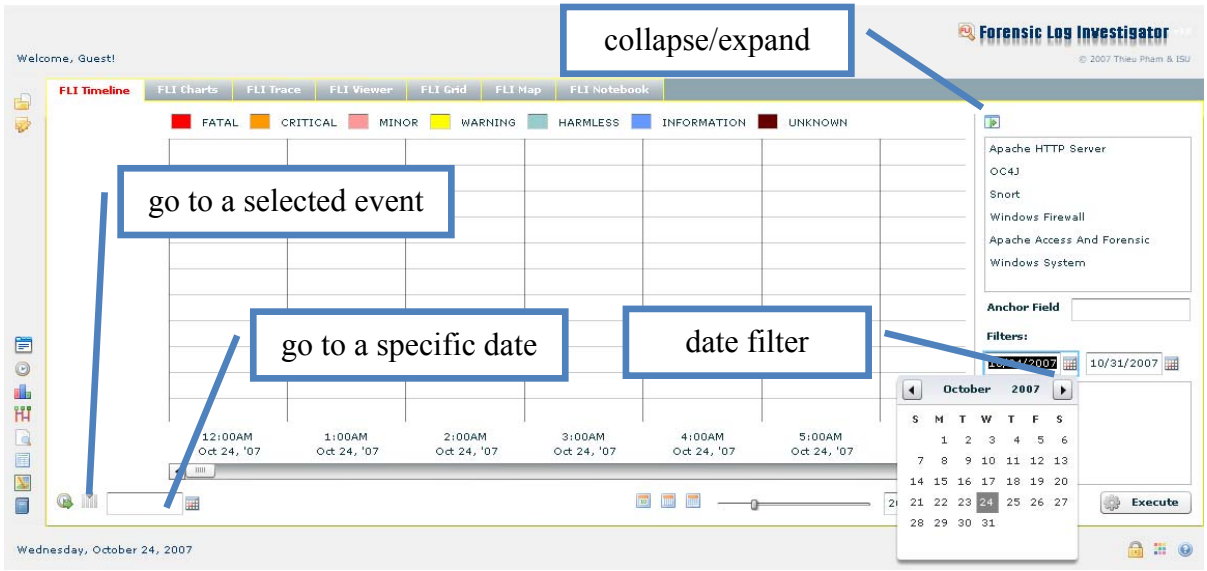

Figure 2.15 FLI Timeline – Layout Cont.

The investigator can filter data within a time range by specifying the start date and end date. The investigator can also jump to a specific date to look for events that happen on that date. To further filter the data, the investigator supplies his/her own XPath expressions.

Figure 2.16 FLI Timeline – Event Color Designation

Since each event's severity level is signified by a color, the investigator can visually skip through the events with low severity levels and focus on those with high severity levels.

To quickly see the details of an event in timeline and bookmark suspicious events, the investigator can do so by hovering the mouse over the event or double-clicking on the event.

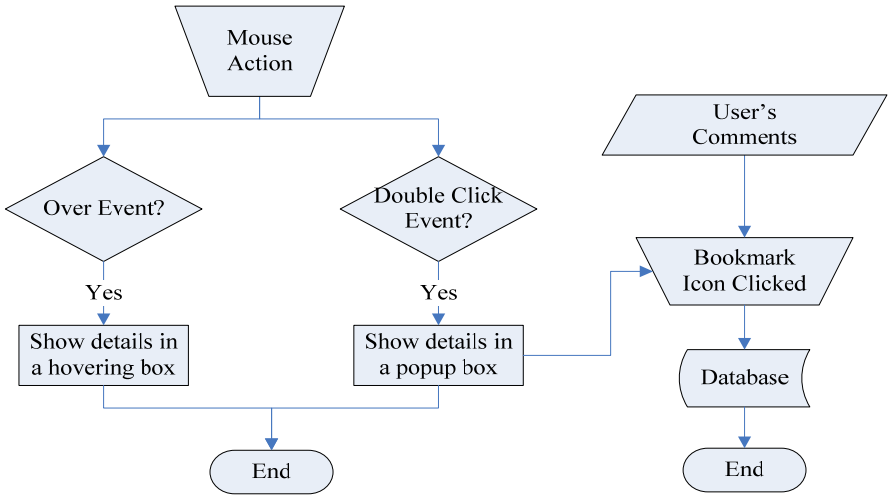

Figure 2.17 Display Event Details Flowchart

Whereas the details hover box will only be displayed when the mouse is over the event, the details pop-up window will stay on the screen until it is closed by the investigator.

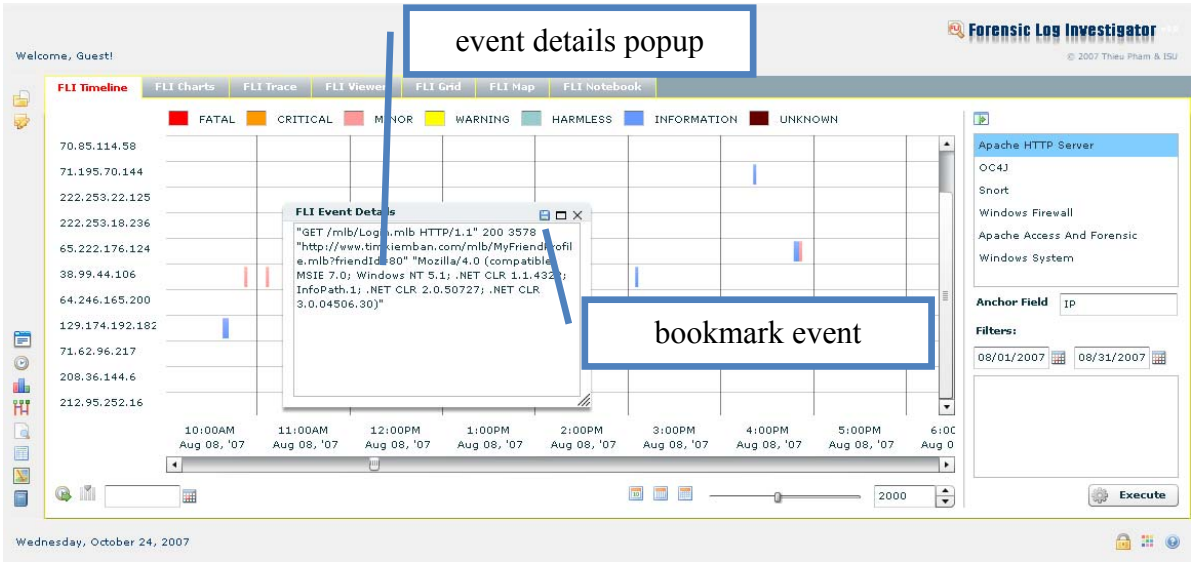

Figure 2.18 FLI Event Details Pop-up Window

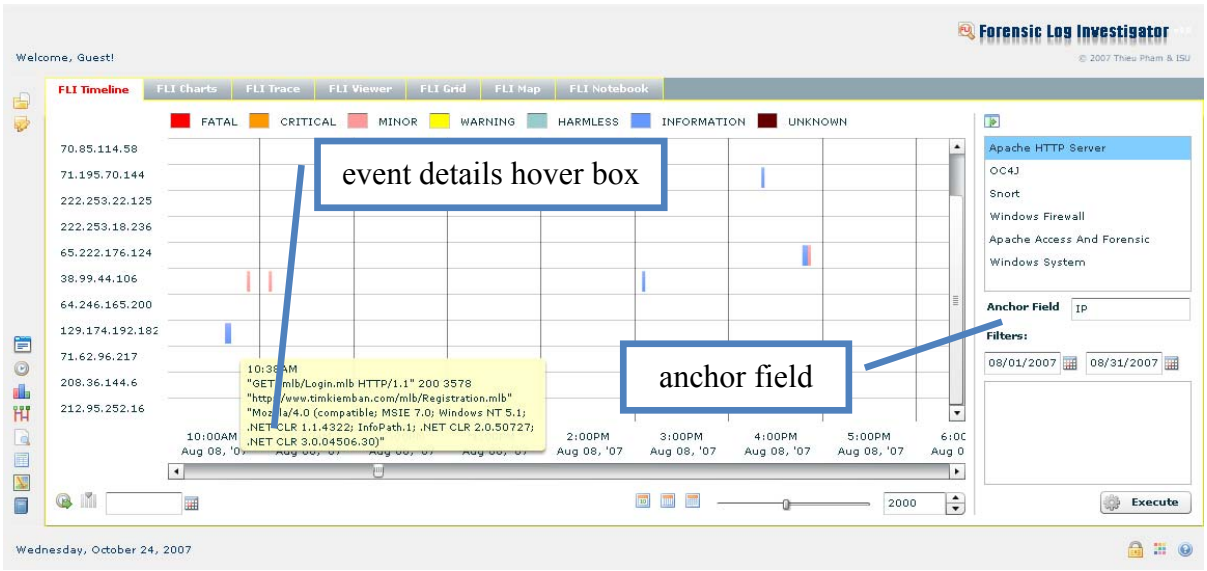

Figure 2.19 FLI Event Details Hover Box

The following figure shows the timeline correlation view. If two or more logs are selected at the same time, the timeline shows the time correlation of the events in those logs. With this, the investigator can see the relationships between the different logs based on time. This helps the investigator reconstruct the sequence of events among the different, disparate logs.

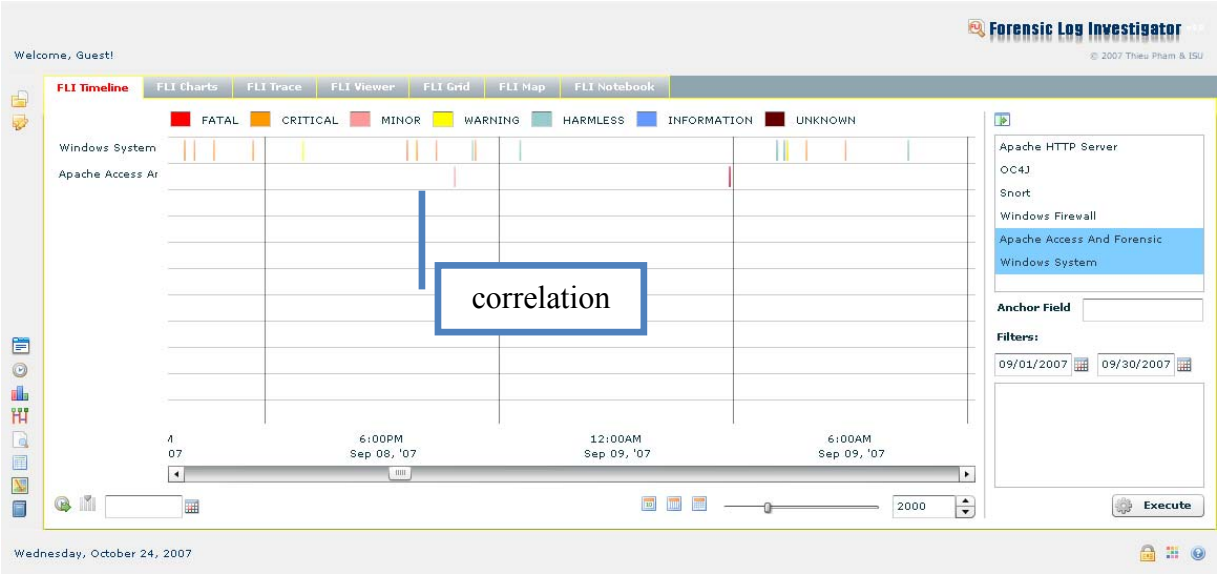

Figure 2.20 FLI Time – Correlation View

"Time lining of computer evidence, when coordinated with other physical events, can provide a broader picture of the sequence and timing of events relating to criminal behavior. [3]"

2.3.4 FLI Charts

We all know how charts can be used to aggregate data and explain statistics effectively. FLI realizes these benefits by providing a number of charts, namely, column chart, bar chart, pie chart, plot chart, line chart, and area chart that the user can use to suffice some analysis and presentation needs.

Depending on the type of a chart, different parameters are taken to generate the chart. This process is defined in the following flowchart.

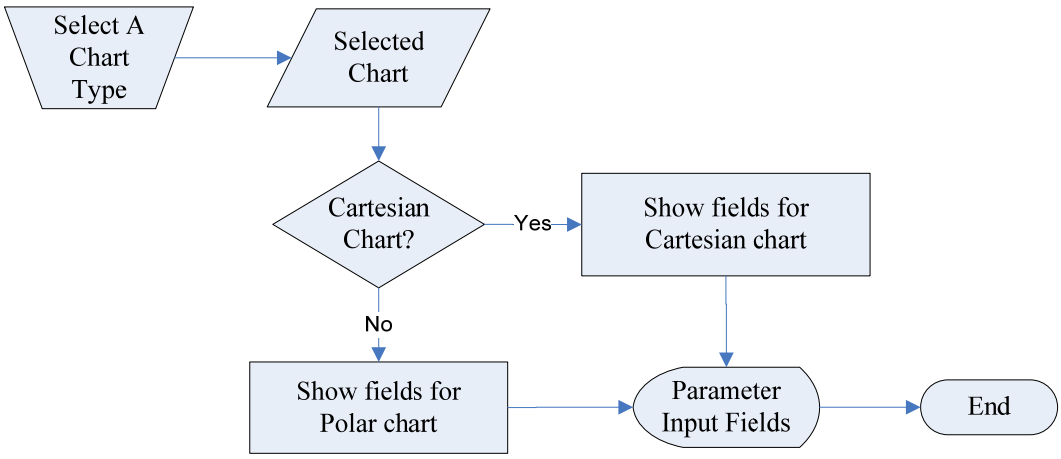

Figure 2.21 FLI Charts - Show Fields Flowchart

Given the required parameters, a chart can be generated. The following flowchart shows how to generate a Cartesian chart.

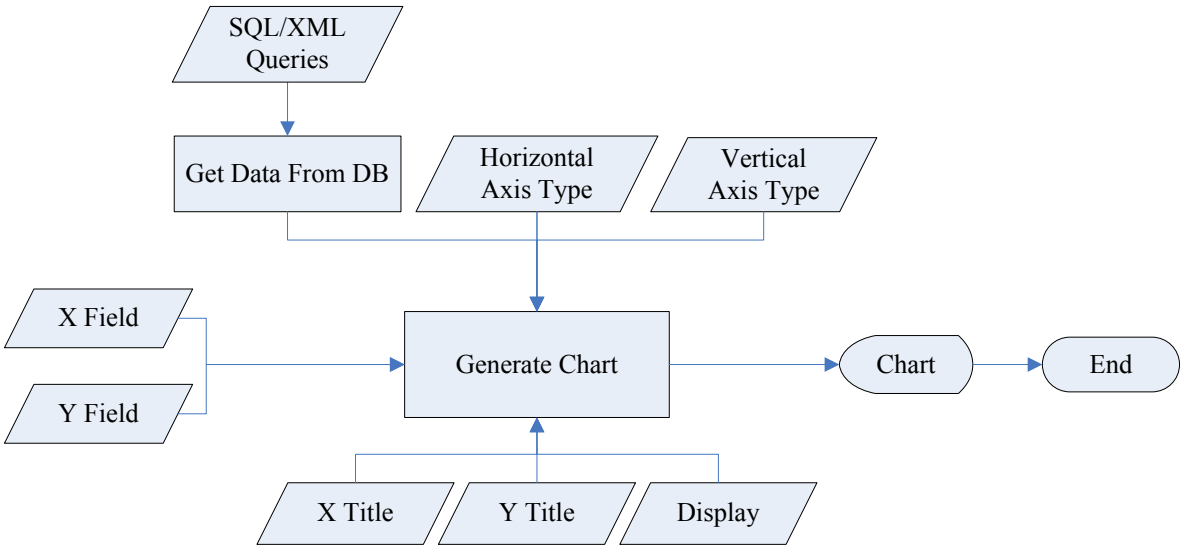

Figure 2.22 FLI Charts – Cartesian Chart Generation Flowchart

The following figures show the Cartesian charts: column chart, bar chart, line chart, plot chart and area chart. The types for the horizontal and vertical axis are category, linear, log, and time. X Field is the field value for the x-axis. Y Field is the field value for the y-axis.

Figure 2.23 FLI Charts - Column Chart

Figure 2.24 FLI Charts – Bar Chart

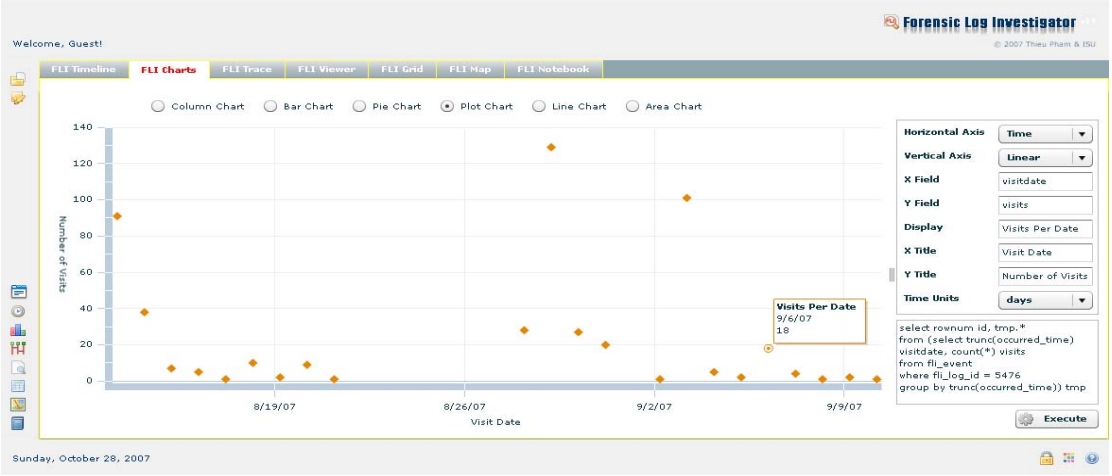

Figure 2.25 FLI Charts – Plot Chart

Figure 2.26 FLI Chart – Line Chart

Figure 2.27 FLI Chart – Area Chart

Figure 2.28 shows the flowchart of how to generate a pie chart.

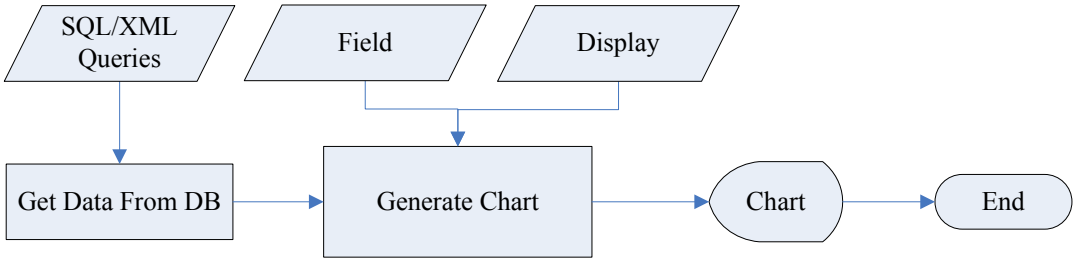

Figure 2.28 FLI Charts - Pie Chart Generation Flowchart

Figure 2.29 shows a pie chart.

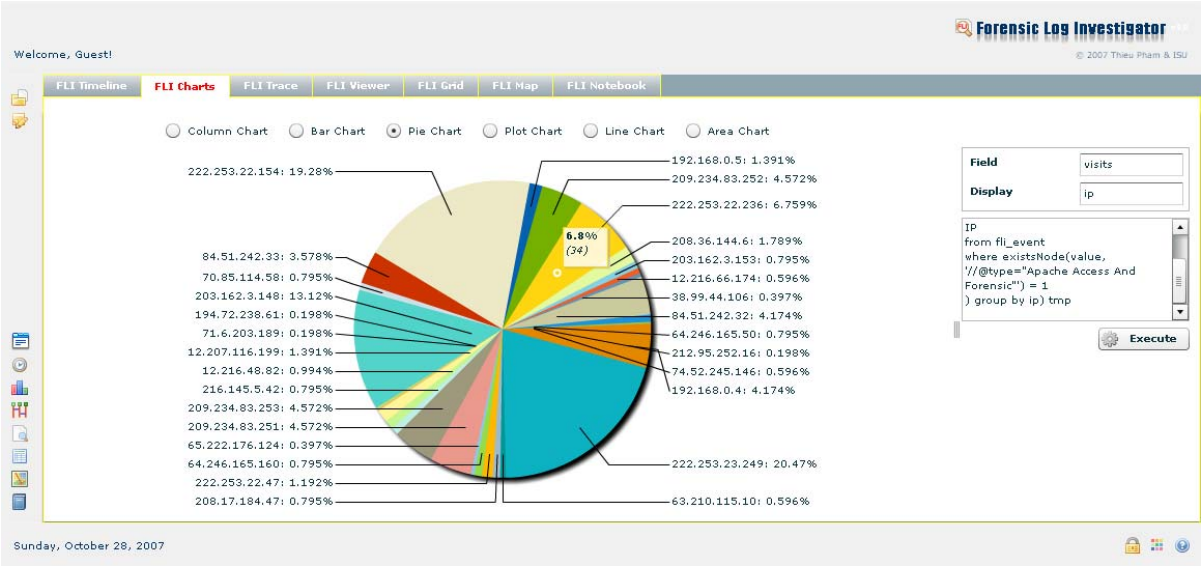

Figure 2.29 FLI Charts - Pie Chart

```
select rownum id, tmp.*
from (select ip,count(*) as visits
      from (select extractvalue(value, '//IP') as IP
            from fli_event
            where
            existsNode(value,'//@type="Apache Access And Forensic"') = 1
      ) group by ip) tmp
```

This query is used to generate the column, bar, and pie charts above. It returns the number of visits per IP address for the "Apache Access and Forensic" log.

```
select rownum id, tmp.*
from (select trunc(occurred_time) visitdate, count(*) visits
      from fli_event
      where existsNode(value, '//@type="Apache Access And Forensic"') = 1
group by trunc(occurred_time)) tmp
```

This query is used to generate the plot, line, and area charts above. It returns the number of visits per date for the "Apache Access and Forensic" log.

2.3.5 FLI Trace

Using FLI Trace, the investigator is able to visually see how the events relate to one another and the nature of those events quickly by the colors of the lines. The investigator can follow the conversation between machines and focus on the communication of a specific number of machines in question or follow the path through a web site that a user has taken. FLI Trace provides the facility to generate those diagrams; however, it is up to the investigator's imagination and experience to tell it appropriately what to generate.

FLI Trace graphically displays the events in either sequence or path diagram and one-hop or multi-hop. One-hop link is, for instance, A → B, A → C, A → D. Multi-hop link is, for instance, A → B, B → C, C → D.

The following chart defines the process to display the fields appropriately to collect different parameters

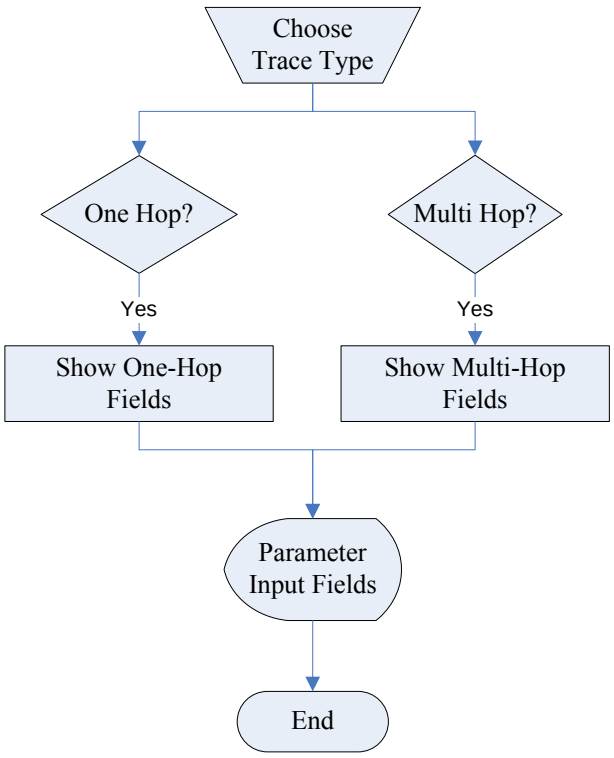

Figure 2.30 FLI Trace – Show Parameter Fields Flowchart

The following flowchart defines the process to generate a one-hop sequence diagram.

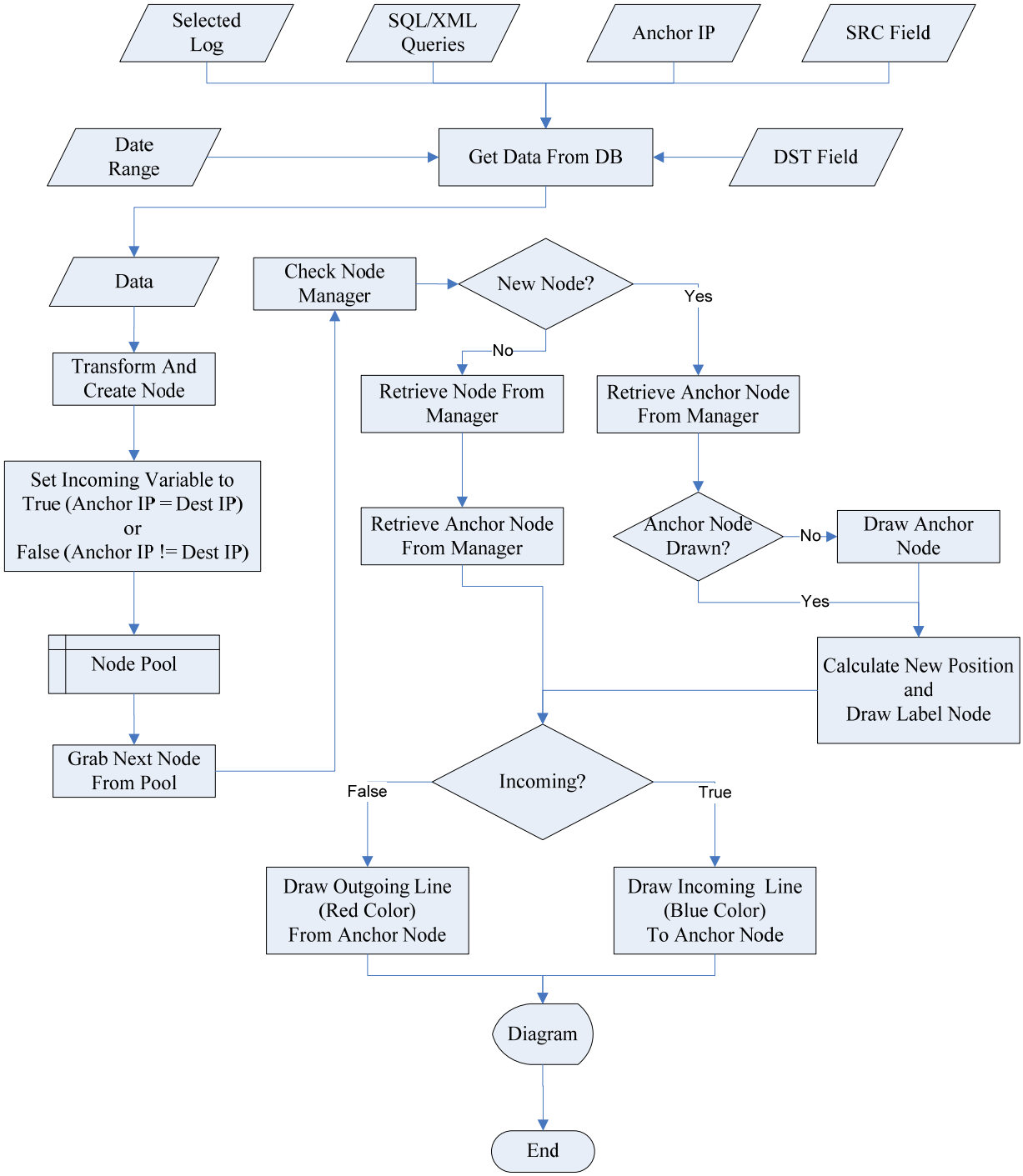

Figure 2.31 FLI Trace - One-hop Diagram Generation Flowchart

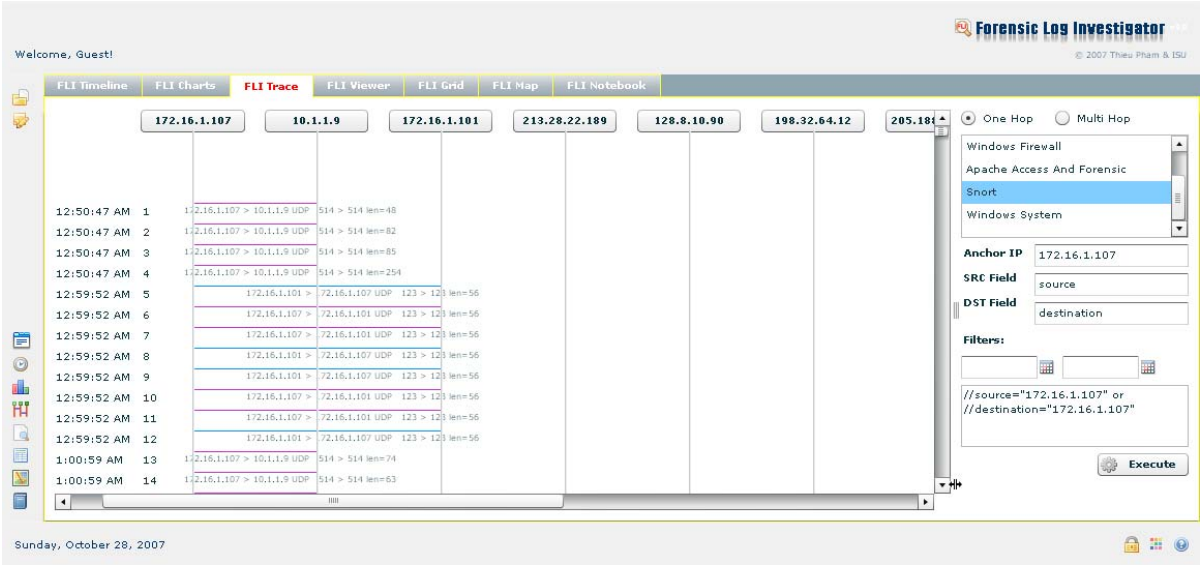

Figure 2.32 FLI Trace - One-hop Sequence Diagram

Figure 2.32 shows a one-hop sequence diagram of Snort. This figure shows the communication between the main machine 172.16.1.107 and the other machines. The communication between the main host and the different hosts is shown in lines of two different colors indicating the directions of the communication, red for outgoing and blue for incoming. A short description of the event is displayed below the line. For a one-hop sequence, the times of the events are shown on the left along with the sequence number.

Anchor IP is the IP of the main host. SRC Field and DST Field are the names of the elements in FLIEvent which contain the source IP and destination IP.

For a multi-hop scenario, besides the sequence diagram, the path diagram is also available. The following flowchart shows the process to determine whether to generate a sequence diagram or path diagram.

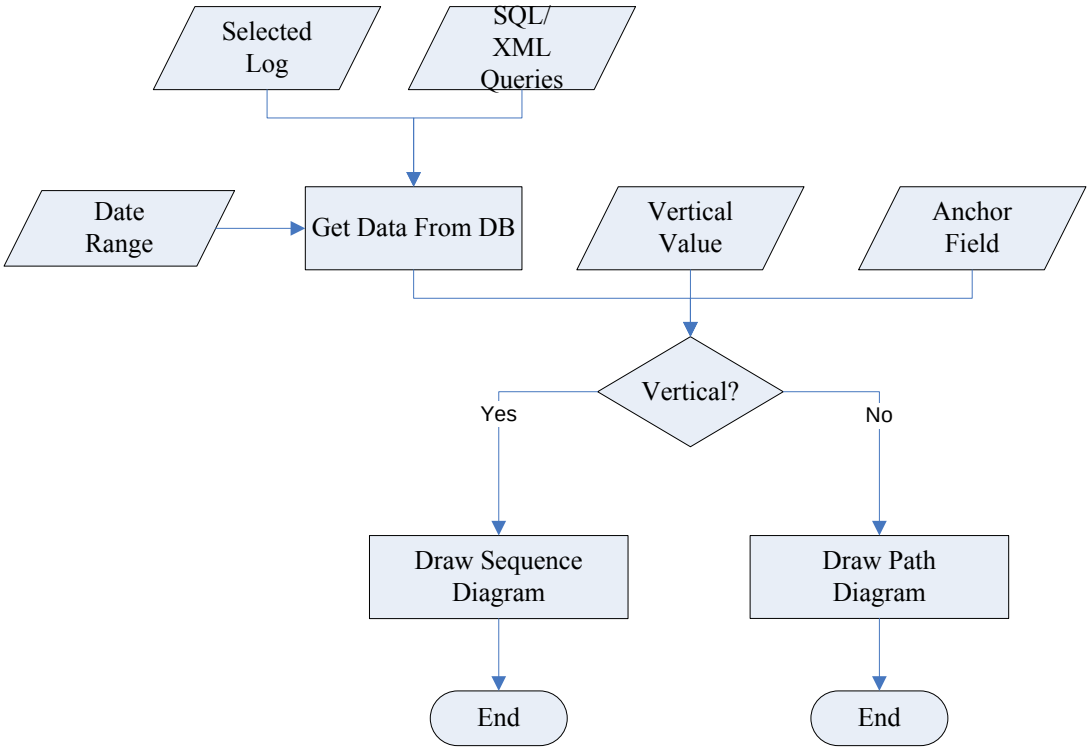

Figure 2.33 FLI Trace - Diagram Type Decision Flowchart

The following flowchart defines the process to generate a multi-hop sequence or path diagram.

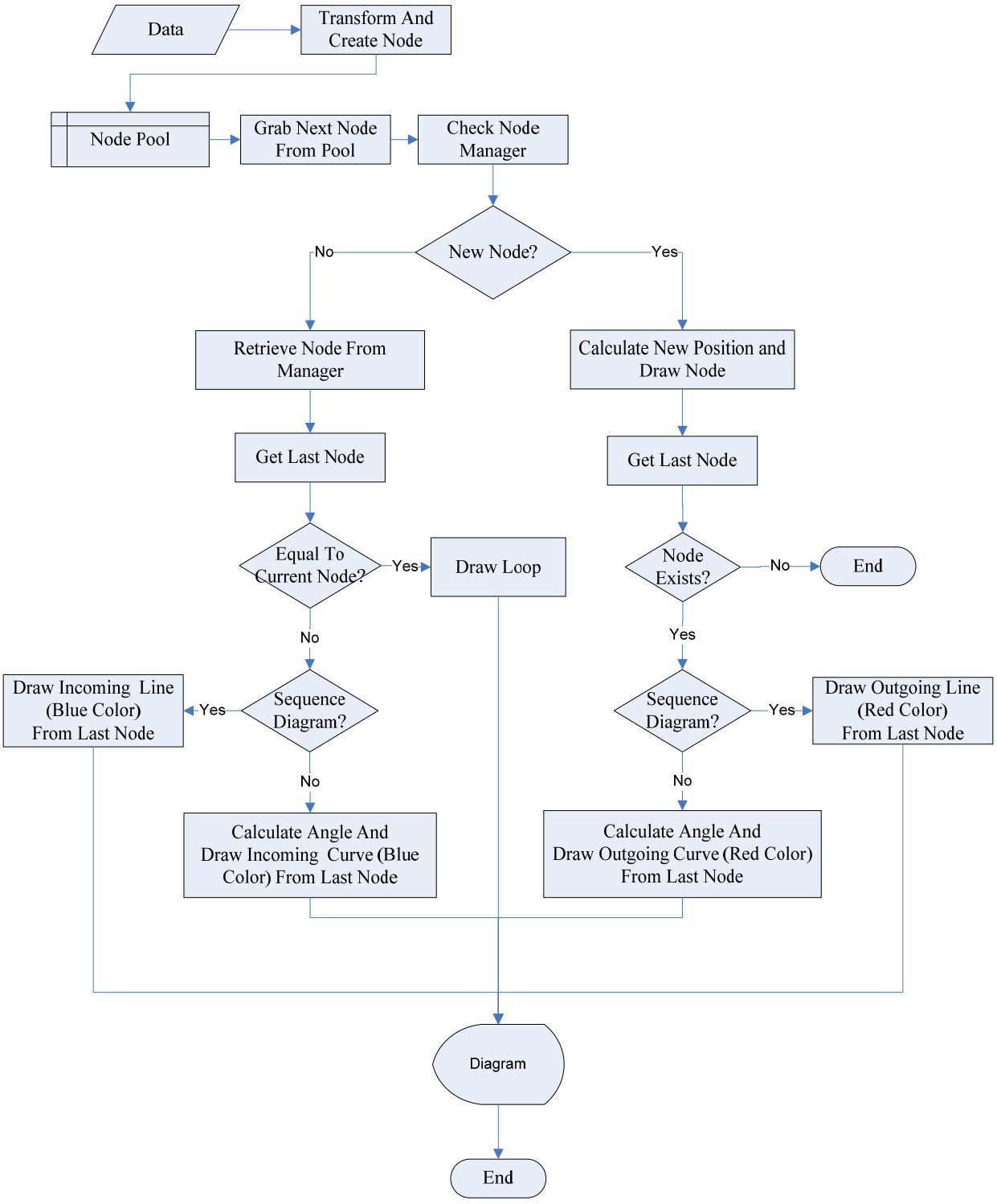

Figure 2.34 FLI Trace - Multi-hop Diagram Generation Flowchart

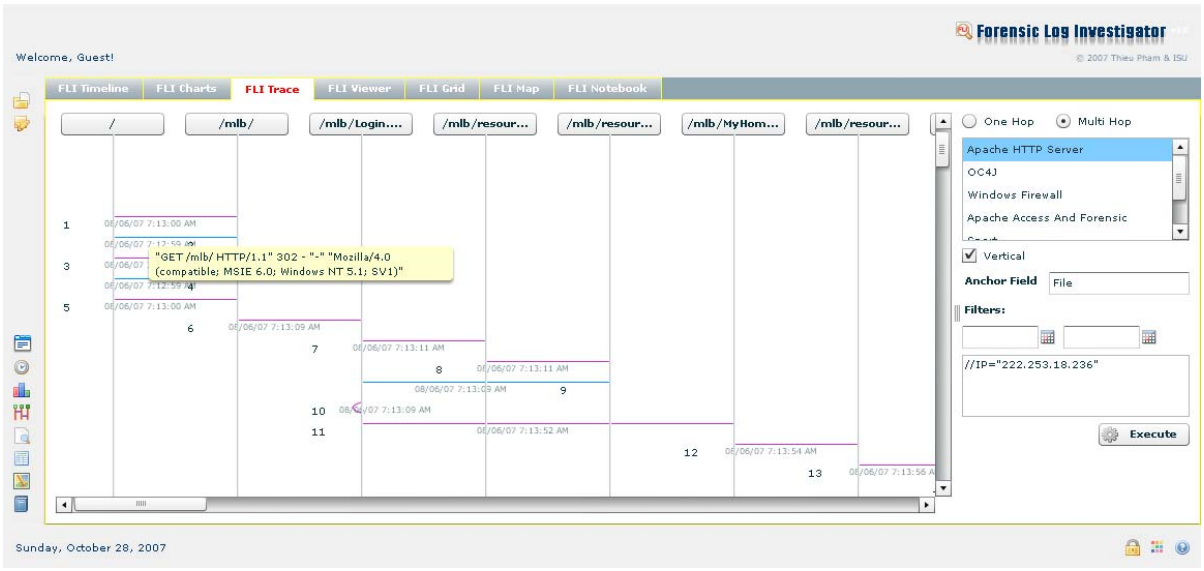

Figure 2.36 FLI Trace - Multi-Hop Sequence Diagram

Figure 2.35 shows the sequence diagram as a user traverses a web site.

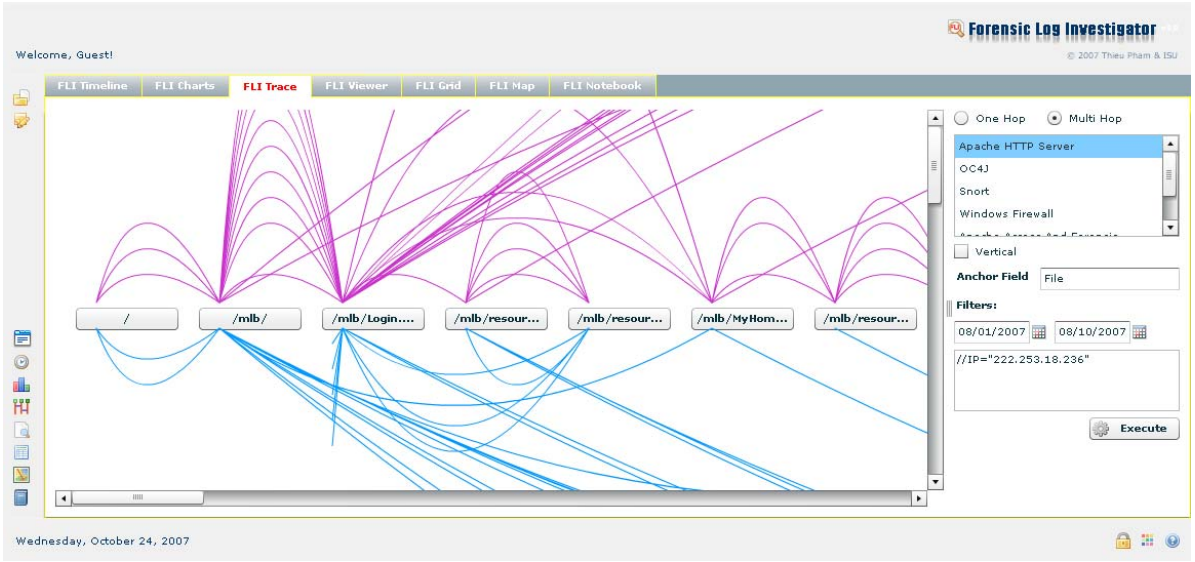

Figure 2.35 FLI Trace - Multi-Hop Path Diagram

Figure 2.36 shows the path and the pages that the user visits. This clearly shows where the user enters and exits and the number of times the user enters or exits a page. Using this, the investigator can trace the path and follow the trail.

Figure 2.37 FLI Trace - Multi-Hop Path Diagram - One Path Focus

Figure 2.37 shows the paths that the user takes from a certain page when that page is chosen. The other unrelated paths to the page are grayed out, focusing the investigator's attention to one area at a time.

2.3.6 FLI Grid

One of the best ways to organize information is in a grid. The investigator can sort and view the events by category. Because each event may have different contents, the biggest technical challenge here is to make the grid flexible enough to dynamically adapt to the different data.

The following flowchart shows how FLI Grid retrieves data from the database.

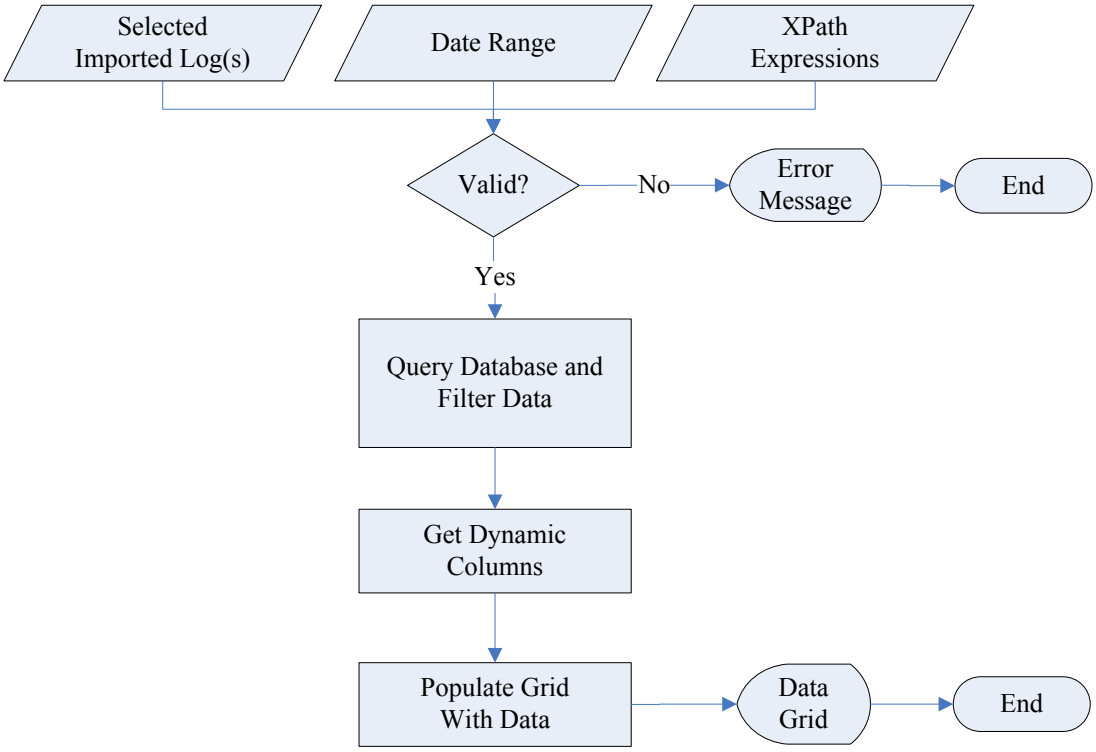

Figure 2.38 FLI Grid - Data Retrieval Flowchart

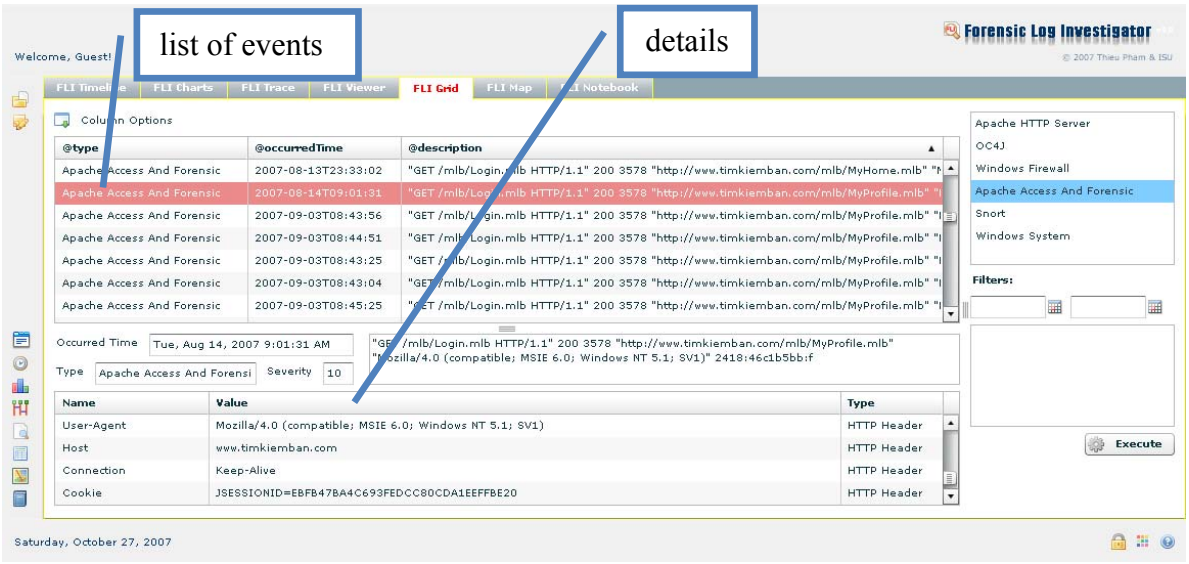

Figure 2.39 FLI Grid - Main View

As the above figure shows, the investigator can easily scroll through the results and examine the details of an event.

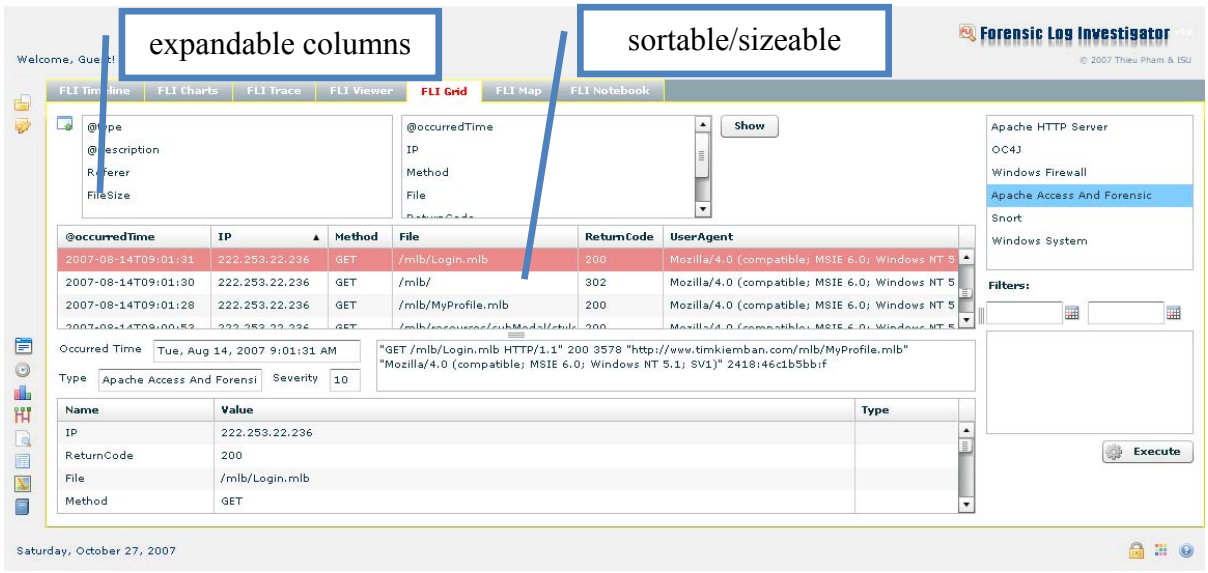

Figure 2.40 FLI Grid – Optional/Dynamic Columns

The columns in FLI Grid are dynamically generated based on the type of the event allowing accommodation for different contents. This has been quite an accomplishment by itself.

2.3.7 FLI Viewer

FLI Viewer allows the investigator to search the log file unprocessed. The benefit here is that the investigator can verify the data confirming the integrity of the data.

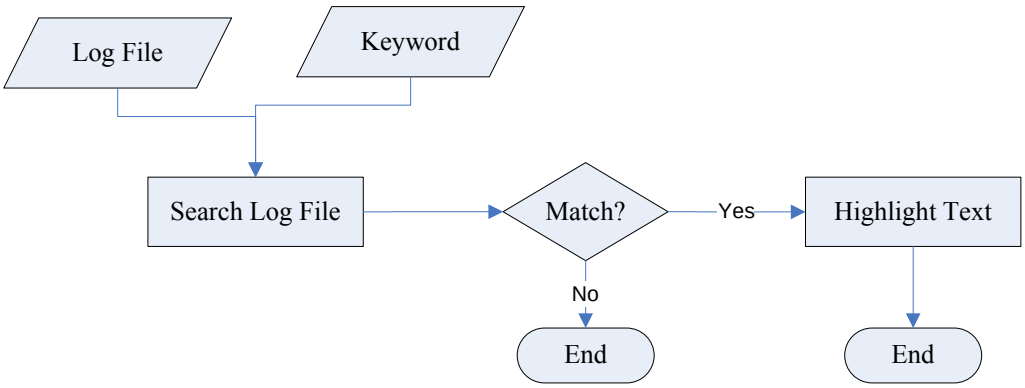

Figure 2.41 FLI Viewer - Search and Highlight Flowchart

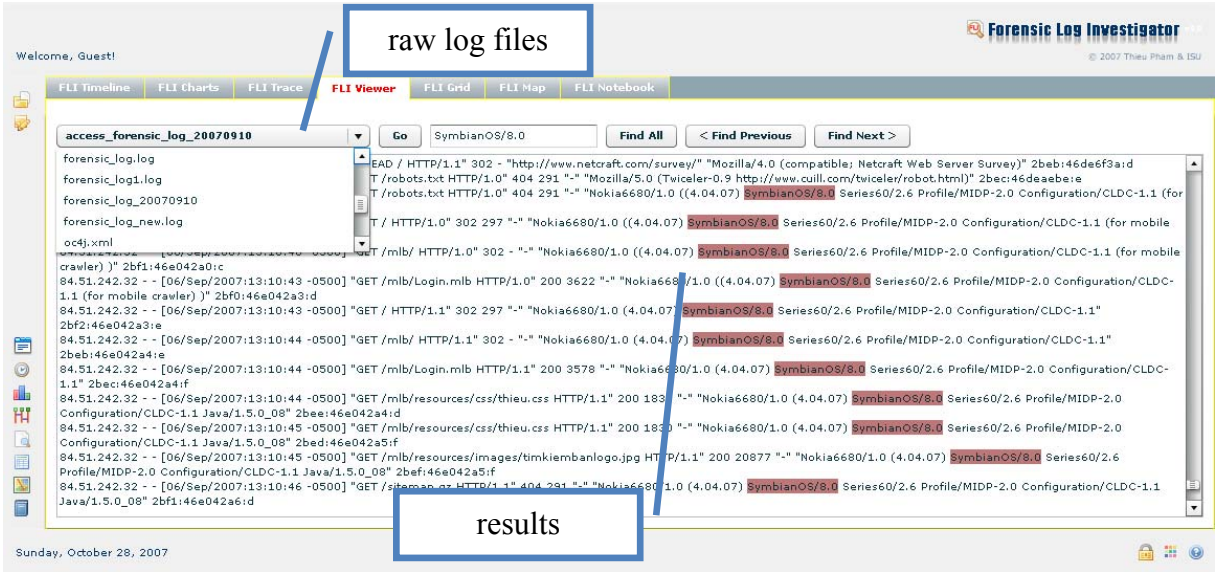

Figure 2.42 FLI Viewer - Find All

The investigator selects an uploaded log file from the repository and searches for keywords. "Find All" will highlight all occurrences in the document.

The investigator can choose “Find Next” to go to the next occurrence or “Find Previous” to go to the previous occurrence.

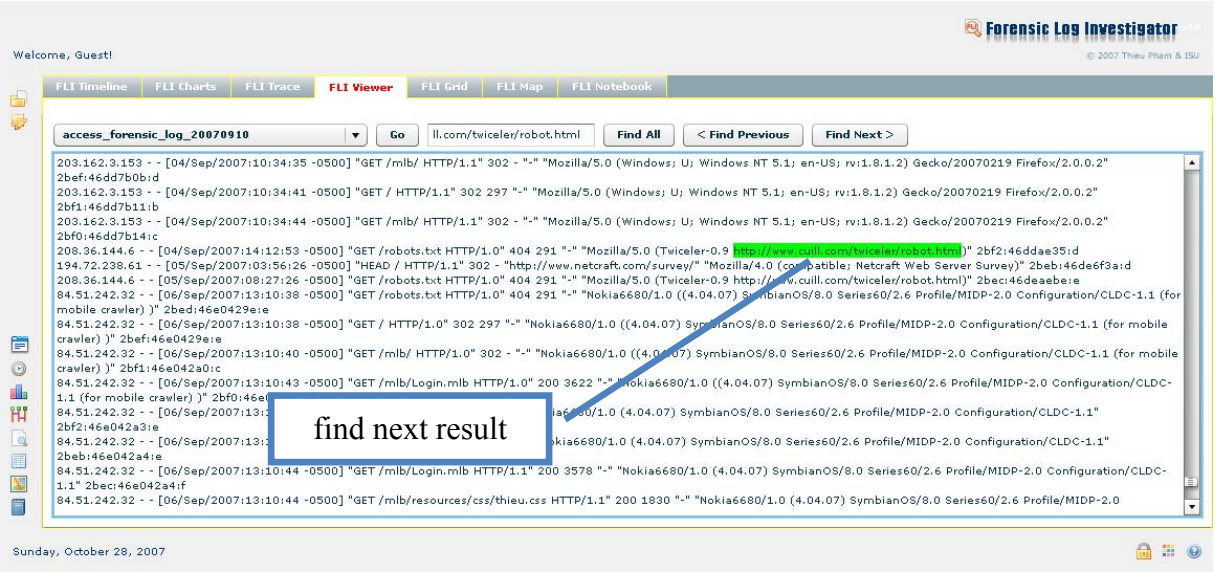

Figure 2.43 FLI Viewer - Single Result

2.3.8 FLI Map

The purpose of FLI Map is to allow investigators to view the physical locations of IP addresses. Using IP geocoding process, an IP can be mapped to a physical location. The intention is to give the investigator the tool to do profiling. Because the Internet is a global entity and some countries are known to be more likely to commit cyber crimes than others, given their locations on a map, the investigator can quickly spot and filter out users based on location. In any investigation, every detail and every clue is important; that little detail may help solve a crime.

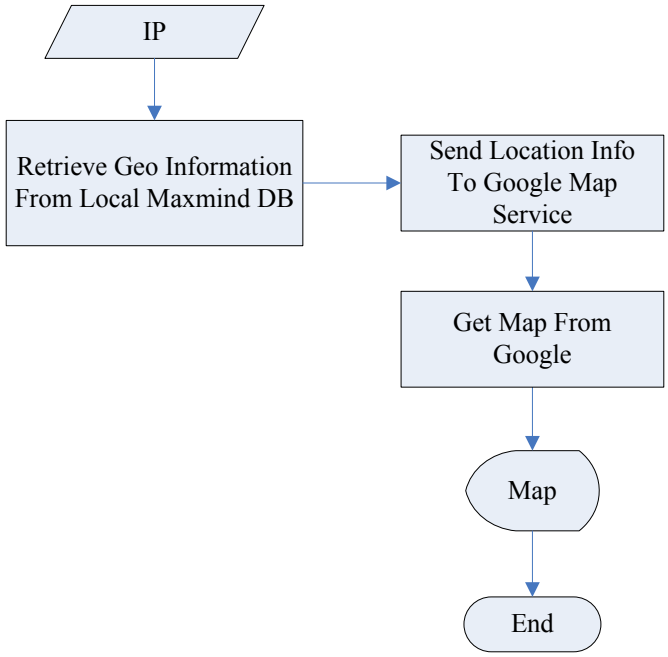

Figure 2.44 FLI Map - IP Map to Location Flowchart

Figure 2.45 shows the locations of some IP addresses, represented by monitor icons. As an icon is clicked, the window pops up and shows the details such as city, state, country, etc., giving the investigator more information about the user.

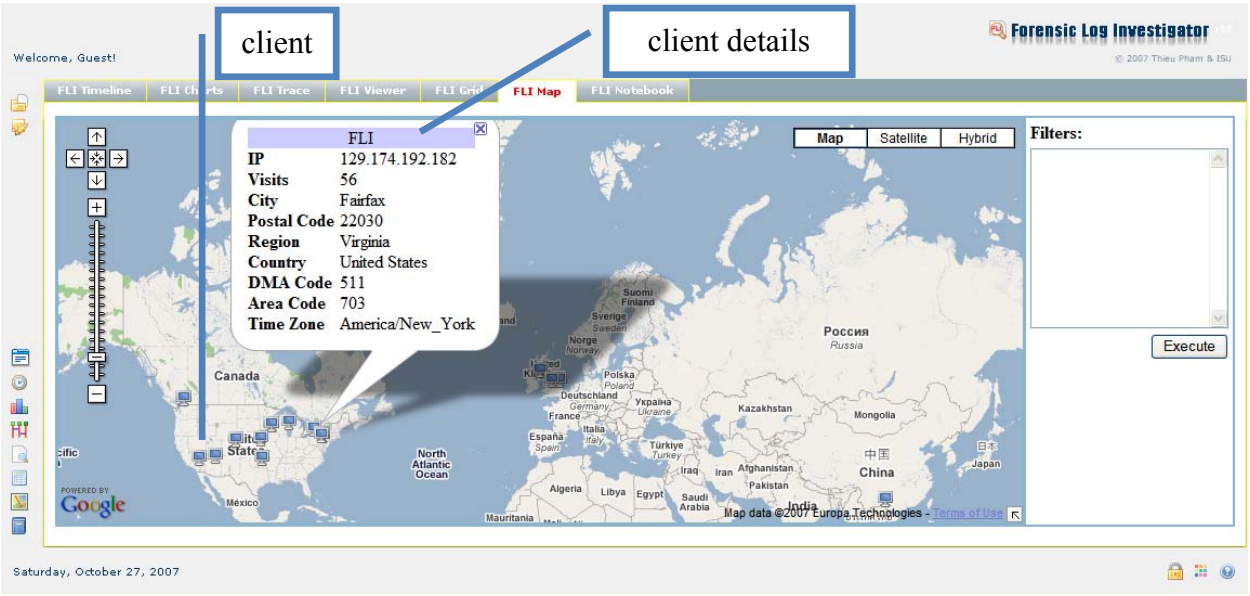

Figure 2.45 FLI Map

The following query is used to generate the result above.

```
select rownum id, tmp.*
from (
  select ip, count(*) as visits
  from (select extractvalue(value, '//IP') as IP
       from fli_event
       where existsNode(value, '//@type="Apache Access And Forensic"') = 1
       ) group by ip) tmp
```

2.3.9 FLI Note & Notebook

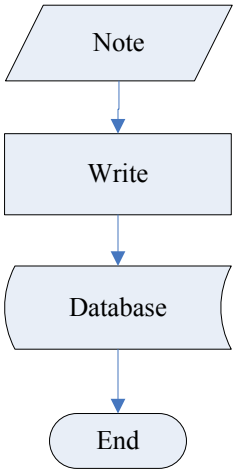

Figure 2.46 FLI Note - Insert New Note Flowchart

FLI Note is available throughout the application. Notes are important in an investigation. As the investigator goes through, he or she has suspicions and ideas of how something happens. As the ideas come, it is important that the investigator jot those down right away. FLI Note allows the investigator to take notes and review them later. The investigator can use this to build up a hypothesis and share his or her findings with other investigators.

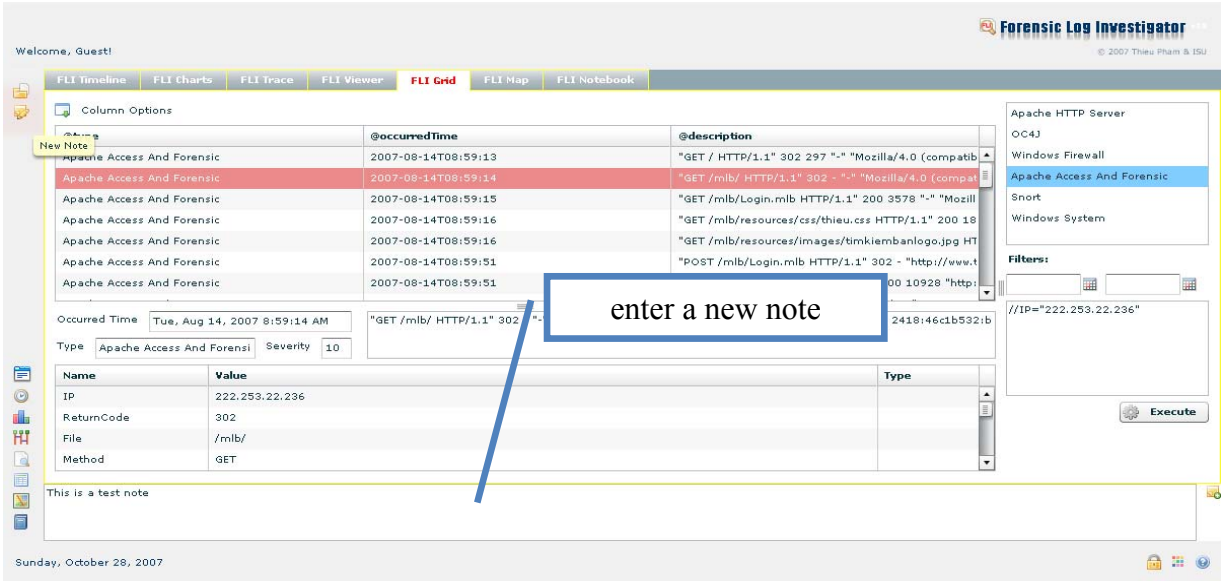

Figure 2.47 FLI Note

Figure 2.48 FLI Notebook

2.3.10 FLI Management & Customization

The investigator manages the imported logs here. The investigator can view the information related to the log.

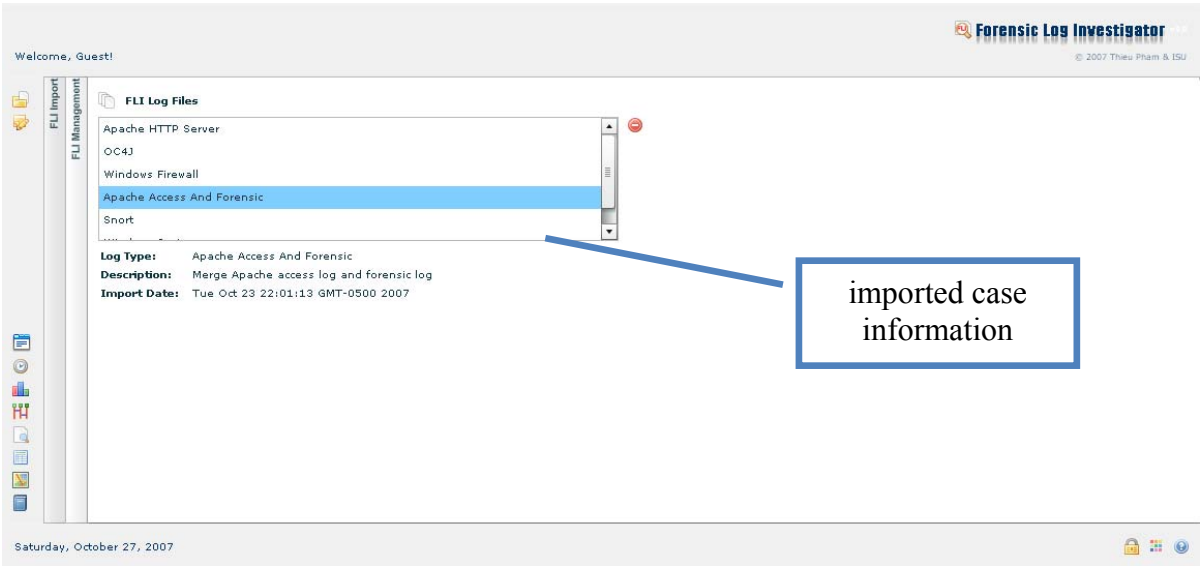

Figure 2.49 FLI Management

A tool is only useful if it is used. Performing an investigation can be tedious and mundane. When the investigator is tired or bored, productivity decreases. Each person has his or her own preferences. FLI provides this feature so that each investigator can customize according his or her own tastes, providing a good look and feel for the investigator.

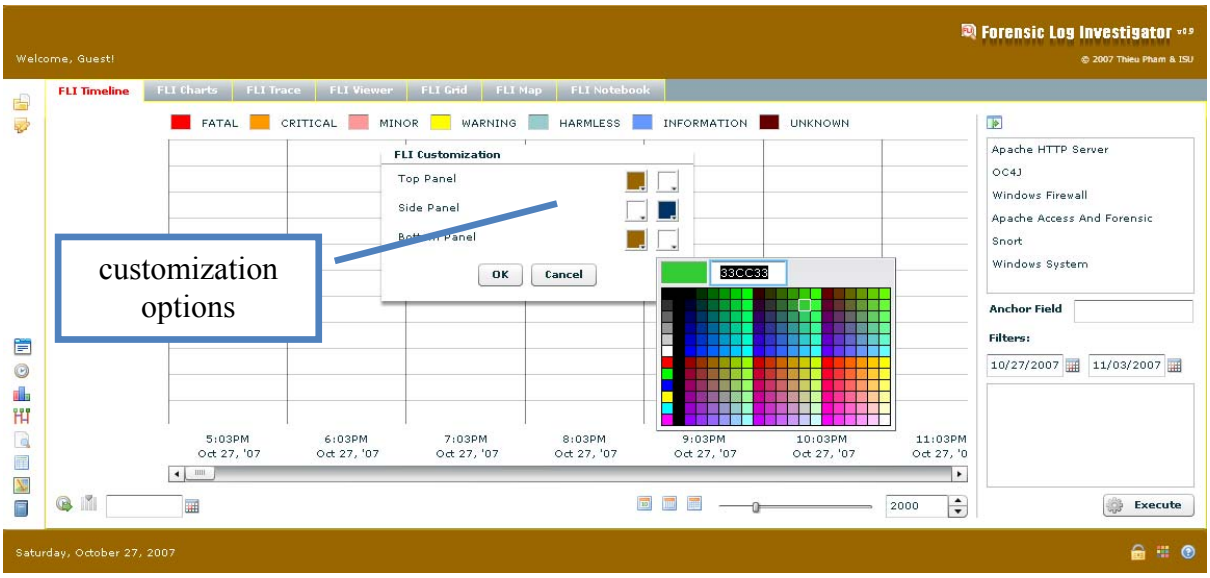

Figure 2.50 FLI Customization

CHAPTER 3. INFRASTRUCTURE

FLI is built upon a solid, powerful, and enterprise infrastructure. FLI uses the latest industry standard technologies leveraging stability, power, and performance.

3.1 Computer Languages

3.1.1 Java/JSP

Java [6] is a high-level object-oriented programming language developed by Sun Microsystems in the early 1990s. Its goal is "compile once, run anywhere." It is widely used and is the force behind many of the existing web applications.

JSP (JavaServer Pages) [7] is Java technology that is used to generate dynamic HTML pages.

3.1.2 ActionScript 3.0

ActionScript [8] is name of a scripting language, based on ECMAScript, used in the development software, especially software which runs in the Adobe Flash Player.

3.1.3 XML

XML (Extensible Markup Language) [9] is a W3C-recommended general-purpose markup language that supports a wide variety of applications. XML languages or 'dialects' are easy to design and to process. It is cross-platform, human-readable, and portable.

3.1.4 MXML

MXML [10] is an XML-based user interface markup language. The acronym stands for "Multimedia eXtensible Markup Language". It allows application developers to develop Rich Internet applications. It is used mainly to declaratively lay-out the interface of applications, and can also be used in conjunction with ActionScript to allow developers to implement complex business logic and rich internet application behaviors.

3.1.5 XPath

XPath [11] is used to address parts of an XML document. In support of this primary purpose, it also provides facilities for manipulation of strings, numbers and Booleans. XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values. XPath operates on the abstract, logical structure of an XML document, rather than its surface syntax. XPath gets its name from its use of a path notation as in URLs for navigating through the hierarchical structure of an XML document. In addition to its use for addressing, XPath is also designed so that it has a natural subset that can be used for matching (testing whether or not a node matches a pattern).

3.1.6 SQL

SQL [12] is a data query language designed for the retrieval and management of data in relational database management systems.

3.1.7 SQL/XML

SQL/XML [13] provides support for using XML in the context of an SQL database system. Because SQL is the standard language for accessing and managing data stored in relational databases, it is natural need for users to have the ability to integrate their XML data into their relational data through the use of SQL facilities. SQL/XML makes it possible to work with XML data in a SQL database using SQL and XPath.

3.1.8 XSD

XSD (XML Schema Definition) [14] is a XML-based language used to describe the structure of an XML document.

3.1.9 XSLT

XSLT (eXtensible Stylesheet Language) [15] is a language for transforming XML documents into other XML documents.

3.1.10 JavaScript

JavaScript [16] is a scripting language used for client-side development.

3.1.11 Regular Expressions

Regular expressions [17] are a way to describe a set of strings based on common characteristics shared by each string in the set. They can be used to search, edit, or manipulate text and data. You must learn a specific syntax to create regular expressions — one that goes beyond the normal syntax of the Java programming language.

3.2 Software Frameworks

3.2.1 Adobe Flex 2.0

Adobe Flex [18] is an umbrella term for a group of technologies to support the development and deployment of rich Internet applications based on their Flash platform. In a multi-tiered model, Flex applications serve as the presentation tier.

3.2.2 Oracle DBMS

Oracle Database [19] is a multithreaded, multi-user SQL database management system that supports relational and XML data. To provide facilities for XML querying and management, they support SQL/XML. FLI uses the free version Oracle Database XE.

3.2.3 JEE (Java Enterprise Edition) Application Server

An application server is a software that runs on a server (hardware) and manages web applications handling communication between the server and client computers. A JEE-

compliant application server [20] runs on a Java platform. JRun [32] is a JEE-compliant and used for this project.

3.2.4 Flex Data Services (FDS)

Adobe Flex Data Services [21] enables developers to build applications that are rich not only in terms of user interface, but also in terms of how data flows between tiers. Deployed as a standard J2EE application, Flex Data Services enhances the client-side Flex framework by providing high-performance connectivity with existing server-side data and business logic. Based on a robust messaging architecture, Flex Data Services integrates with existing common middleware and provides services that automatically synchronize data between client and server, add support for real-time data push and publish/subscribe messaging, and enable collaborative and occasionally disconnected applications.

3.2.5 Adobe Flash Player

Adobe Flash Player [22] is a widely distributed multimedia and application player created and distributed by Adobe Systems. Flash Player is the virtual machine used to run the Flash file.

3.2.6 Spring API

Spring [23] is an inversion of control (a.k.a. dependency injection) container for Java.

3.2.7 Hibernate API

Hibernate [24] is an object-relational mapping (ORM) solution for the Java language.

3.2.8 DOM4J API

DOM4J [25] is a Java framework for working with XML documents.

3.2.9 Xalan-Java API

Xalan-Java [26] is an XSLT processor for transforming XML documents into HTML, text, or other XML document types. It implements XSL Transformations (XSLT) Version 1.0 and XML Path Language (XPath) Version 1.0 and can be used from the command line, in an applet or a servlet, or as a module in other program.

3.2.10 Xerces-Java API

Xerces-Java [27] is a high performance, fully compliant XML parser.

3.2.11 GLA API

GLA [28] allows you to process application log files and transform their contents into the Common Base Event (CBE) format. The Common Base Event specification prescribes a common format for logging, management, problem determination, and autonomic computing.

3.2.12 JnetStream API

JnetStream [29] is a Java API for working with captured network packets.

3.2.13 Maxmind GeoIP API

GeoIPAPI [30] is an open source API to allow look ups of Maxmind GeoIP database. GeoIP database contains country, region, city, postal code, area code associated with an IP. FLI uses GeoLite City [33] which is the free version and available for download.

3.2.14 Google Maps API

Google Maps API [31] lets you embed Google Maps in your own web pages with JavaScript. The API provides a number of utilities for manipulating maps (just like on the http://maps.google.com web page) and adding content to the map through a variety of services, allowing you to create robust maps applications on your website.

CHAPTER 4. SUMMARY

In this thesis, the design and implementation of FLI have been proposed and implemented. The other important part, included but cannot be presented here, is the source code of FLI. FLI is a fully functional tool that has achieved the following:

- Flexible and extensible way to work with different log formats and contents
- Intuitive, interactive, rich, and user-friendly interface
- Visualization of data for pattern recognition and discovery through colors and diagrams
- Powerful searching and filtering capabilities
- Easy sorting and organization of data
- Efficient handling and management of large amount of log data
- Global accessibility for users
- Profiling of web users by mapping IP addresses to physical locations
- Visual statistical analysis through charts
- Bookkeeping for review and reporting

With the many features provided in FLI, it is a handy tool that helps investigators be productive. The FLI log parsing framework resolves the problem of different log formats and contents by providing the adapters and the FLI log format FLIEvent. Using an Oracle RDMS and built-in pagination feature of FDS, FLI can efficiently handle and manage large amount of data. With powerful searching and filtering capabilities, the investigator can find and retrieve the information in question quickly. Once found, the information can be easily organized and sorted to help the investigator narrow down and focus on a set of events. Visualization can be used not only for finding patterns but also for presentation which is an important part of an investigation. FLI is a web application, making it easy to maintain and accessible to all users from anywhere.

As always, things are more easily said than done. In the process of researching and implementing FLI, design and technical challenges have had to be met and overcome. It has taken a great deal of effort to have formulated the idea and figured out how to best approach it. There has been a lot of time spent researching and studying the best technologies available and how to optimally use them. As shown in chapter 3, there are quite a few technologies used to make FLI possible. Each one is a challenge in itself to learn and configure. When it comes to the actual implementation, there have been many challenges also. Users do not see the code beneath, but it is what makes it possible. It is where the most time is spent. Another difficulty encountered in this research is the acquisition of sample log files and understanding of the different log formats.

Certainly, because of the time constraint and the scope of this research, there are things that may be missing or left unaddressed. The great thing about FLI is that it is built on a solid foundation. Everyone knows how crucial it is to have a good foundation. FLI has laid down a good foundation which makes it a lot easier for other features to be added and built upon.

The overall objective of this research is to create a tool that is useful for investigators in conducting investigations – saving time, money, and effort. FLI is our result, and therefore, this research has accomplished that objective.

FLI, after all, is just a tool. Indeed, it is a great tool; nevertheless, it is up to the experience and the skills of the investigator to make the best use out of it.

For future work, more testing will be done and more features can be added. Data mining techniques can be integrated with FLI to give investigators more information and different views of data that will help in analysis. Voice and video can also be added to FLI to promote collaboration and sharing information, ideas and expertise.

APPENDIX A. FLI SOURCE CODE

Please contact Professor Yong Guan at guan@iastate.edu

APPENDIX B. CBE TO FLIEVENT XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output omit-xml-declaration="yes"/>
   <xsl:template match="/">
            <xsl:element name="FLIEvent">
                  <xsl:attribute name="type">
                        <xsl:value-of
                              select="//sourceComponentId/@component"/>
                  </xsl:attribute>
                  <xsl:attribute name="occurredTime">
                        <xsl:value-of
                              select="/CommonBaseEvent/@creationTime"/>
                  </xsl:attribute>
                  <xsl:attribute name="description">
                        <xsl:value-of select="/CommonBaseEvent/@msg"/>
                  </xsl:attribute>
                  <xsl:attribute name="severity">
                        <xsl:value-of
                              select="/CommonBaseEvent/@severity"/>
                  </xsl:attribute>
                  <xsl:attribute name="key">
                        <xsl:value-of select="/CommonBaseEvent/@version"/>
                  </xsl:attribute>
                  <xsl:for-each select="//extendedDataElements">
                        <xsl:variable name="extendedName" select="@name"/>
                        <xsl:element name="{$extendedName}">
                              <xsl:value-of select="./values/text()"/>
                        </xsl:element>
                  </xsl:for-each>
            </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

APPENDIX C. FLIEVENT MERGE XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" indent="yes"/>
   <xsl:variable name="b"
      select="document('C:/data/forensic_log_20070910.out')"/>
   <xsl:template match="/FLIEventSet">
      <xsl:element name="FLIEventSet">
         <xsl:for-each select="FLIEvent">
            <xsl:variable name="key">
               <xsl:value-of select="@key"/>
            </xsl:variable>
            <xsl:element name="FLIEvent">
                <xsl:attribute name="type">
                        Apache Access And Forensic
                      </xsl:attribute>
                <xsl:attribute name="occurredTime">
                    <xsl:value-of select="@occurredTime"/>
                </xsl:attribute>
                <xsl:attribute name="description">
                    <xsl:value-of select="@description"/>
                </xsl:attribute>
                <xsl:attribute name="severity">
                    <xsl:value-of select="@severity"/>
                </xsl:attribute>
               <xsl:copy-of select="child::*"/>
               <xsl:element name="Extension">
                    <xsl:attribute name="type">
                        <xsl:text>HTTP Header</xsl:text>
                    </xsl:attribute>
                    <xsl:attribute name="description">
                        <xsl:text>Forensic Log</xsl:text>
                    </xsl:attribute>
                    <xsl:copy-of
                        select="$b/FLIEventSet/FLIEvent[@key=$key]/child::
                        *"/>
                </xsl:element>
            </xsl:element>
         </xsl:for-each>
      </xsl:element>
   </xsl:template>
</xsl:stylesheet>
```

APPENDIX D. FLI GLA CINGULAR ADAPTER

```
<?xml version="1.0" encoding="UTF-8"?>
<adapter:Adapter
  xmlns:adapter="http://www.eclipse.org/hyades/schema/Adapter.xsd"
  xmlns:cc=
  "http://www.eclipse.org/hyades/schema/ComponentConfiguration.xsd"
  xmlns:ex="http://www.eclipse.org/hyades/schema/Extractor.xsd"
  xmlns:fmt="http://www.eclipse.org/hyades/schema/Formatter.xsd"
  xmlns:hga="http://www.eclipse.org/hyades/schema/Context.xsd"
  xmlns:op="http://www.eclipse.org/hyades/schema/Outputter.xsd"
  xmlns:parser="http://www.eclipse.org/hyades/schema/Parser.xsd"
  xmlns:pu="http://www.eclipse.org/hyades/schema/ProcessUnit.xsd"
  xmlns:sensor="http://www.eclipse.org/hyades/schema/Sensor.xsd">
  <hga:Contexts>
    <hga:Context
      description="Context Instance for the current component"
      executableClass=
      "org.eclipse.hyades.logging.adapter.impl.BasicContext"
      implementationCreationDate="2007-09-22T20:31:00"
      implementationVersion="1.0" loggingLevel="30"
      name="Basic Context Implementation" role="context"
      roleCreationDate="2007-09-22T20:31:00"
      roleVersion="1.0"
      uniqueID="N4360A01097411DC8000853B7A54A13E">
      <hga:Component
        description="Operating System file sensor"
        executableClass="org.eclipse.hyades.logging.adapter.
            sensors.SingleOSFileSensor"
        implementationCreationDate="2007-09-22T20:31:00"
        implementationVersion="1.0" loggingLevel="30"
        name="OS File Sensor" role="sensor"
        roleCreationDate="2007-09-22T20:31:00"
        roleVersion="1.0"
        uniqueID="N4360A03097411DC8000853B7A54A13E" />
      <hga:Component
        description="This extractor uses regular expression
            patterns to identify record delimiters"
        executableClass="org.eclipse.hyades.logging.
            adapter.extractors.RegularExpressionExtractor"
        implementationCreationDate="2007-09-22T20:31:00"
        implementationVersion="1.0" loggingLevel="30"
        name="Regular Expression Extractor"
        role="messageExtractor"
        roleCreationDate="2007-09-22T20:31:00"
        roleVersion="1.0"
        uniqueID="N4360A05097411DC8000853B7A54A13E" />
      <hga:Component description="Regular expression parser"
        executableClass="org.eclipse.hyades.logging.adapter
            .parsers.Parser"
        implementationCreationDate="2007-09-22T20:31:00"
        implementationVersion="1.0" loggingLevel="30"
        name="Generic Parser" role="parser"
        roleCreationDate="2007-09-22T20:31:00"
```

```
      roleVersion="1.0"
      uniqueID="N4360A06097411DC8000853B7A54A13E" />
    <hga:Component description="CBE Formatter"
      executableClass="org.eclipse.hyades.logging.adapter
          .formatters.CBEFormatter"
      implementationCreationDate="2007-09-22T20:31:00"
      implementationVersion="1.0" loggingLevel="30"
      name="CBE Formatter" role="formatter"
      roleCreationDate="2007-09-22T20:31:00"
      roleVersion="1.0"
      uniqueID="N4360A0A097411DC8000853B7A54A13E" />
    <hga:Component description="FLI Database Outputter"
      executableClass="fli.adapter.FLIEventOutputter"
      implementationCreationDate="2007-09-22T20:31:07"
      implementationVersion="1.0" loggingLevel="30"
      name="FLI Outputter" role="outputter"
      roleCreationDate="2007-09-22T20:31:07"
      roleVersion="1.0"
      uniqueID="NE2AEC44097411DC8000853B7A54A13E" />
  </hga:Context>
</hga:Contexts>
<cc:Configuration
  description="The component level configurations for this Adaptor"
  uniqueID="N4360A00097411DC8000853B7A54A13E">
  <cc:ContextInstance charset=""
    continuousOperation="false"
    description="Context Instance for the current component"
    isoCountryCode="" isoLanguageCode=""
    maximumIdleTime="" pauseInterval=""
    uniqueID="N4360A01097411DC8000853B7A54A13E">
    <cc:Sensor description="A single file sensor"
      uniqueID="N4360A03097411DC8000853B7A54A13E"
      confidenceBufferSize="1024" maximumBlocking="5"
      type="SingleFileSensor">
      <pu:Property propertyName="directory"
        propertyValue="C:\test " />
      <pu:Property propertyName="fileName"
        propertyValue="phonelog.txt" />
      <sensor:SingleFileSensor
        directory="C:\test "
        fileName="phonelog.txt" />
    </cc:Sensor>
    <ex:Extractor
      description="This extractor uses regular expression
          patterns to identify record delimiters"
      uniqueID="N4360A05097411DC8000853B7A54A13E"
      containsLineBreaks="true" endPattern="$"
      includeEndPattern="true" includeStartPattern="true"
      lineBreakSymbol="" replaceLineBreaks="false"
      startPattern="^" />
    <cc:Parser separatorToken=";"
      uniqueID="N4360A06097411DC8000853B7A54A13E">
      <parser:RuleElement
        index="N4360A07097411DC8000853B7A54A13E"
        isChildChoice="false" isRequiredByParent="false"
```

```
name="CommonBaseEvent">
<parser:RuleElement
  index="N6D7882F097B11DC8000853B7A54A13E"
  name="sourceComponentId">
  <parser:RuleAttribute defaultValue="CINGULAR"
    index="N6D78830097B11DC8000853B7A54A13E"
    isRequiredByParent="false" name="component"
    usePreviousMatchSubstitutionAsDefault="false" />
  <parser:RuleAttribute
    index="N6D78831097B11DC8000853B7A54A13E"
    isRequiredByParent="false"
    name="componentIdType"
    usePreviousMatchSubstitutionAsDefault="false">
    <SubstitutionRule match="^(.*)"
      substitute="replace with our message text"
      useBuiltInFunction="false" />
  </parser:RuleAttribute>
  <parser:RuleAttribute
    index="N6D78832097B11DC8000853B7A54A13E"
    isRequiredByParent="false"
    name="componentType"
    usePreviousMatchSubstitutionAsDefault="false">
    <SubstitutionRule match="^(.*)"
      substitute="replace with our message text"
      useBuiltInFunction="false" />
  </parser:RuleAttribute>
  <parser:RuleAttribute
    index="N6D78833097B11DC8000853B7A54A13E"
    isRequiredByParent="false" name="location"
    usePreviousMatchSubstitutionAsDefault="false">
    <SubstitutionRule match="^(.*)"
      substitute="replace with our message text"
      useBuiltInFunction="false" />
  </parser:RuleAttribute>
  <parser:RuleAttribute
    index="N6D78834097B11DC8000853B7A54A13E"
    isRequiredByParent="false" name="locationType"
    usePreviousMatchSubstitutionAsDefault="false">
    <SubstitutionRule match="^(.*)"
      substitute="replace with our message text"
      useBuiltInFunction="false" />
  </parser:RuleAttribute>
  <parser:RuleAttribute
    index="N3EBC079098211DC8000853B7A54A13E"
    name="subComponent"
    usePreviousMatchSubstitutionAsDefault="false">
    <SubstitutionRule useBuiltInFunction="false" />
  </parser:RuleAttribute>
</parser:RuleElement>
<parser:RuleElement
  index="NDB12BF7097411DC8000853B7A54A13E"
  name="situation">
  <parser:RuleElement
    index="NF121B8D097411DC8000853B7A54A13E"
    name="RequestSituation">
```

```
      <parser:RuleAttribute
        index="NF121B8E097411DC8000853B7A54A13E"
        isRequiredByParent="false"
        name="reasoningScope"
        usePreviousMatchSubstitutionAsDefault="false">
        <SubstitutionRule match="^(.*)"
          substitute="replace with our message text"
          useBuiltInFunction="false" />
      </parser:RuleAttribute>
      <parser:RuleAttribute
        index="NF121B8F097411DC8000853B7A54A13E"
        isRequiredByParent="false"
        name="situationQualifier"
        usePreviousMatchSubstitutionAsDefault="false">
        <SubstitutionRule match="^(.*)"
          substitute="replace with our message text"
          useBuiltInFunction="false" />
      </parser:RuleAttribute>
      <parser:RuleAttribute
        index="NF121B90097411DC8000853B7A54A13E"
        isRequiredByParent="false"
        name="successDisposition"
        usePreviousMatchSubstitutionAsDefault="false">
        <SubstitutionRule match="^(.*)"
          substitute="replace with our message text"
          useBuiltInFunction="false" />
      </parser:RuleAttribute>
    </parser:RuleElement>
  </parser:RuleElement>
  <parser:RuleElement
    index="N925A1B3097911DC8000853B7A54A13E"
    name="extendedDataElements">
    <parser:RuleAttribute defaultValue="CALL_TO"
      index="N925A1B4097911DC8000853B7A54A13E"
      isRequiredByParent="false" name="name"
      usePreviousMatchSubstitutionAsDefault="false" />
    <parser:RuleAttribute
      index="N0B80660097911DC8000853B7A54A13E"
      name="values"
      usePreviousMatchSubstitutionAsDefault="false">
      <SubstitutionRule match="(.*)" positions="4"
        substitute="$1" useBuiltInFunction="false" />
    </parser:RuleAttribute>
    <parser:RuleAttribute
      index="N10C19B6097A11DC8000853B7A54A13E"
      name="type"
      usePreviousMatchSubstitutionAsDefault="false" />
  </parser:RuleElement>
  <parser:RuleElement
    index="N0A4F003097911DC8000853B7A54A13E"
    name="extendedDataElements">
    <parser:RuleAttribute
      defaultValue="NUMBER_CALLED"
      index="N0A4F004097911DC8000853B7A54A13E"
      isRequiredByParent="false" name="name"
```

```
    usePreviousMatchSubstitutionAsDefault="false" />
  <parser:RuleAttribute
    index="ND4AB330097911DC8000853B7A54A13E"
    name="values"
    usePreviousMatchSubstitutionAsDefault="false">
    <SubstitutionRule match="(.*)" positions="5"
      substitute="$1" useBuiltInFunction="false" />
  </parser:RuleAttribute>
  <parser:RuleAttribute
    index="N35DE3A6097A11DC8000853B7A54A13E"
    name="type"
    usePreviousMatchSubstitutionAsDefault="false" />
</parser:RuleElement>
<parser:RuleElement
  index="N26A3C03097911DC8000853B7A54A13E"
  name="extendedDataElements">
  <parser:RuleAttribute defaultValue="RATE"
    index="N26A3C04097911DC8000853B7A54A13E"
    isRequiredByParent="false" name="name"
    usePreviousMatchSubstitutionAsDefault="false" />
  <parser:RuleAttribute
    index="NBA28880097A11DC8000853B7A54A13E"
    name="values"
    usePreviousMatchSubstitutionAsDefault="false">
    <SubstitutionRule match="(.*)" positions="6"
      substitute="$1" useBuiltInFunction="false" />
  </parser:RuleAttribute>
  <parser:RuleAttribute
    index="N93259F6097A11DC8000853B7A54A13E"
    name="type"
    usePreviousMatchSubstitutionAsDefault="false" />
</parser:RuleElement>
<parser:RuleElement
  index="N05A58D3098111DC8000853B7A54A13E"
  name="extendedDataElements">
  <parser:RuleAttribute defaultValue="MINUTES"
    index="N05A58D4098111DC8000853B7A54A13E"
    isRequiredByParent="false" name="name"
    usePreviousMatchSubstitutionAsDefault="false" />
  <parser:RuleAttribute
    index="NA648990098111DC8000853B7A54A13E"
    name="values"
    usePreviousMatchSubstitutionAsDefault="false">
    <SubstitutionRule match="(.*)" positions="7"
      substitute="$1" useBuiltInFunction="false" />
  </parser:RuleAttribute>
</parser:RuleElement>
<parser:RuleAttribute defaultValue="EMPTY"
  index="N4360A08097411DC8000853B7A54A13E"
  isRequiredByParent="false" name="creationTime"
  usePreviousMatchSubstitutionAsDefault="false">
  <SubstitutionRule match="(.*)@@(.*)@@(.*)"
    positions="1@@2@@3" substitute="$1/2007 $2 $3"
    timeFormat="MM/dd/yyyy HH:mm "
    useBuiltInFunction="false" />
```

```
            </parser:RuleAttribute>
            <parser:RuleAttribute defaultValue="EMPTY"
              index="N6D1C00E097411DC8000853B7A54A13E"
              name="msg"
              usePreviousMatchSubstitutionAsDefault="false">
              <SubstitutionRule match="(.*)@@(.*)"
                positions="4@@5" substitute="$1 $2"
                useBuiltInFunction="false" />
            </parser:RuleAttribute>
          </parser:RuleElement>
        </cc:Parser>
        <fmt:Formatter description="CBE Formatter"
          uniqueID="N4360A0A097411DC8000853B7A54A13E" />
        <cc:Outputter
          description="CommonBaseEvent Standard Out Outputter"
          uniqueID="NE2AEC44097411DC8000853B7A54A13E"
          type="StandardOutOutputter">
          <op:StandardOutOutputterType />
        </cc:Outputter>
      </cc:ContextInstance>
    </cc:Configuration>
  </adapter:Adapter>
```

BIBLIOGRAPHY

[1] Palmer, Gary. A Road Map for Digital Forensic Research. Technical Report DTRT0010-01, DFRWS, November 2001. Report from the First Digital Forensic Research Workshop (DFRWS).

[2] Carrier, Brian. Defining Digital Forensics Examination and Analysis Tools Using Abstraction Layers. International Journal of Digital Evidence. Vol 1, Issue 4, Winter 2003.

[3] Hosmer, Chet. Time-Lining Computer Evidence. Available at http://www.wetstonetech.com/f/timelining.pdf. Retrieved November 5, 2007.

[4] Brian D. Carrier and Eugene H. Spafford. Defining event reconstruction of digital crime scenes. CERIAS Tech Report 2004-37.

[5] Schneier, Bruce; Kelsey, John. Secure Audit Logs to Support Computer Forensics. ACM Transactions on Information and System Security. 1999.

[6] Java. http://java.sun.com/. Retrieved October 24, 2007.

[7] JSP. http://java.sun.com/products/jsp/. Retrieved October 24, 2007.

[8] ActionScript. http://www.adobe.com/devnet/actionscript/. Retrieved October 24, 2007.

[9] XML. http://www.w3.org/XML/. Retrieved October 24, 2007.

[10] MXML. http://www.adobe.com/devnet/flex/articles/paradigm.html. Retrieved October 24, 2007.

[11] XPath. http://www.w3.org/TR/xpath. Retrieved October 24, 2007.

[12] SQL. http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/toc.htm. Retrieved October 24, 2007.

[13] SQL/XML. http://www.oracle.com/technology/tech/xml/xquery/sqlxml/index.html. Retrieved October 24, 2007.

[14] XSD. http://www.w3.org/XML/Schema. Retrieved October 24, 2007.

[15] XSLT. http://www.w3.org/TR/xslt. Retrieved October 24, 2007.

[16] JavaScript. http://www.w3schools.com/js/default.asp. Retrieved October 24, 2007.

[17] Regular Expressions. http://java.sun.com/docs/books/tutorial/essential/regex/. Retrieved October 24, 2007.

[18] Adobe Flex 2. http://www.adobe.com/products/flex/. Retrieved October 24, 2007.

[19] Oracle RMDS. http://www.oracle.com/technology/products/database/xe/index.html. Retrieved October 24, 2007.

[20] J2EE. http://java.sun.com/javaee/. Retrieved October 24, 2007.

[21] Flex Data Services. http://www.adobe.com/products/flex/dataservices/. Retrieved October 24, 2007.

[22] Adobe Flash Player. http://www.adobe.com/products/flashplayer/. Retrieved October 24, 2007.

[23] Spring. http://www.springframework.org/. Retrieved October 24, 2007.

[24] Hibernate. http://www.hibernate.org/. Retrieved October 24, 2007.

[25] DOM4J. http://www.dom4j.org/. Retrieved October 24, 2007.

[26] Xalan-Java. http://xml.apache.org/xalan-j/. Retrieved October 24, 2007.

[27] Xerces-Java. http://xerces.apache.org/xerces2-j/index.html. Retrieved October 24, 2007.
[28] Generic Log Adapter. http://www.eclipse.org/tptp/home/downloads/installguide/gla_44/gla_install.html. Retrieved October 24, 2007.
[29] Jnetstream. http://jnetstream.sourceforge.net/. Retrieved October 24, 2007.
[30] GeoIP API. http://www.maxmind.com/app/api. Retrieved October 24, 2007.
[31] Google Maps. http://www.google.com/apis/maps/. Retrieved October 24, 2007.
[32] JRun 4. http://www.adobe.com/products/jrun/. Retrieved October 24, 2007.
[33] GeoLite City. http://www.maxmind.com/app/ip-location. Retrieved October 24, 2007.
[34] FlexLib Project. http://code.google.com/p/flexlib/. Retrieved October 24, 2007.
[35] Snort Log File. The Honeynet Project. http://www.honeynet.org/. Retrieved October 24, 2007.
[36] Icons. http://www.famfamfam.com/lab/icons/mint/. Retrieved October 24, 2007.
[37] Yasinsac, Alec; Manzano, Yanet. Policies to Enhance Computer and Network Forensics in Proceedings of the 2001 IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY, 5-6 June, 2001.
[38] Walker, Cornell. Computer Forensics: Bringing the Evidence to Court. Available at http://www.infosecwriters.com/text_resources/pdf/Computer_Forensics_to_Court.pdf . Retrieved November 5, 2007.
[39] Ryan, Daniel; Shpantzer, Gal. Legal Aspects of Digital Forensics. The George Washington University, 2000.
[40] NIST. General Test Methodology for Computer Forensic Tools. V. 1.9. November 2001
[41] Segal, Ory. Web Application Forensics: The Uncharted Territory. Sanctum Security Group. Available at http://www.cgisecurity.com/lib/WhitePaper_Forensics.pdf. Retrieved November 5, 2007.

ACKNOWLEDGEMENT

I would like to take this opportunity to express my thanks, first and foremost, to God through whom all things are possible and to those who have helped me with various aspects of conducting research and the writing of this thesis.

I would like to thank Dr. Yong Guan for his guidance, insights and support throughout this research and the writing of this thesis. I would also like to thank my committee members for their time and efforts: Dr. Ying Cai and Dr. Steve Russell. I would especially like to thank Dr. Tien Nguyen as the last-minute substitute for Dr. Steve Russell. I would additionally like to thank CISCO Inc. for the partial financial support and the flexible work schedule and co-workers and friends for understanding and encouragement.

Finally, without these people, my life would not have turned out the way it is. I would like thank my family for all the support, love, and encouragement — my parents who have loved, cared, and sacrificed so much for me throughout my life; my brothers and sisters who have endured me and challenged me to do better.

This work was partially supported by NSF under grants No. DUE-0313837, CNS-0644238, and CNS-0626822.