

Design and Implementation of
Zeitline: a Forensic Timeline
Editor

Florian Buccholz

Courtney Falk

CERIAS, Purdue University

What is It?

- Forensic timeline editor
 - Focuses on event reconstruction such as netForensics' product without additional agent processes
 - Does not handle evidence retrieval like SleuthKit, FTK, et. al.
- Implemented in Java with a Swing GUI
- Just recently left alpha stage and entered beta development

Terms to Know

- Atomic event - discrete time event generated from a source file
- Complex event - collection of atomic events or also other complex events
- Source - file from which events were generated
- Timeline - hierarchy of events structured within a single complex event

Basic Functionality

- Deleting, copying, cutting, and pasting
- Saving/loading projects
- Searching/finding a keyword within a time range if desired
- Querying/filtering events based on a keyword and/or a time range

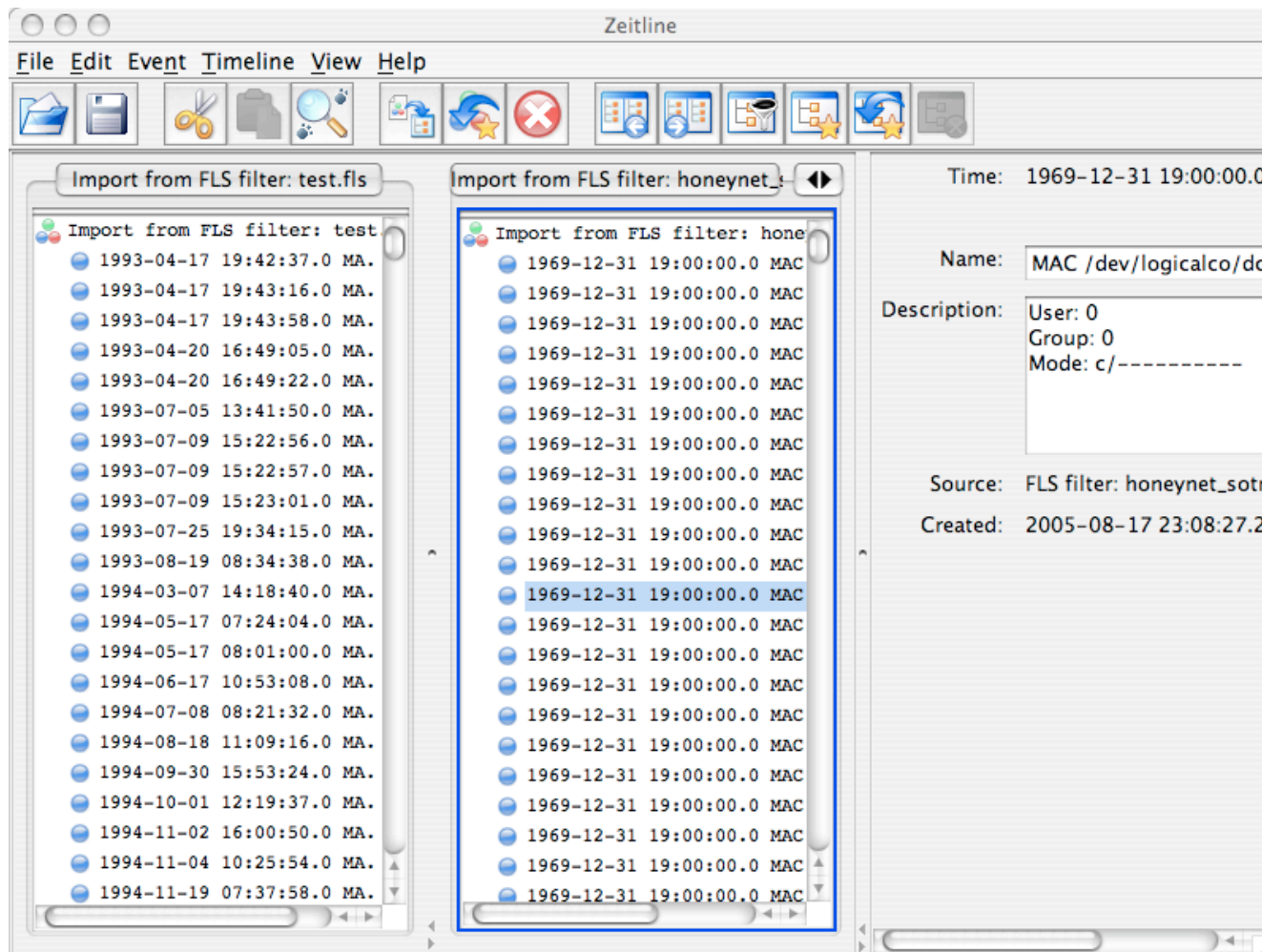
Forensic Integrity Design Considerations

- Orphan timeline - collection of all “deleted” events
- Deleting - all or none with events where either all events from a source are used or none
- Cutting - cut items are stored in a buffer, which is always saved with the project or purged to the orphan timeline if new items are copied/cut

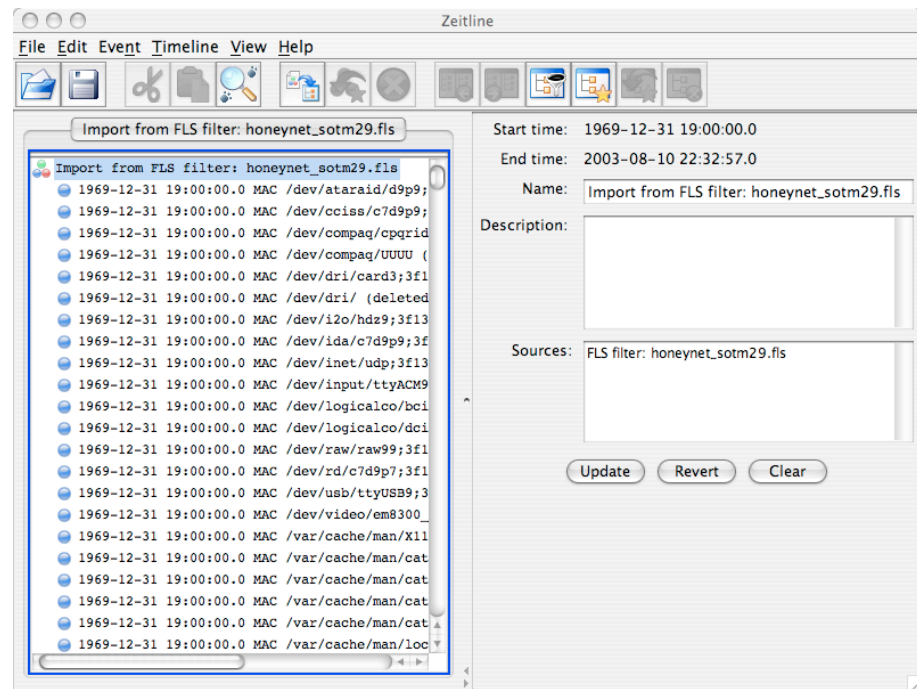
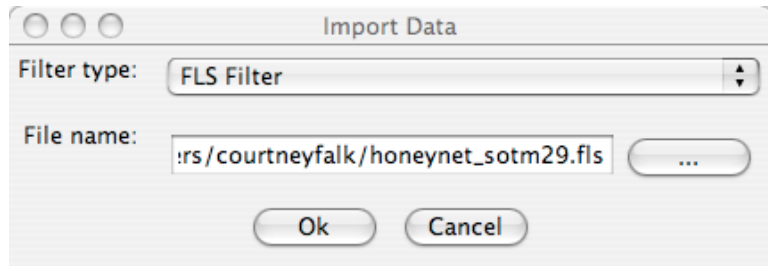
Performance

- Need to handle tens of thousands of elements
- `java.util.TreeSet` provided $O(n^2)$ build time
- Custom Adelson-Velksii-Landis (AVL) balanced search tree gives $O(n \log n)$ build time
- 86k events in 46.6 seconds, or 15k in 9.7s
- *~57%* of processing time is graphical rendering

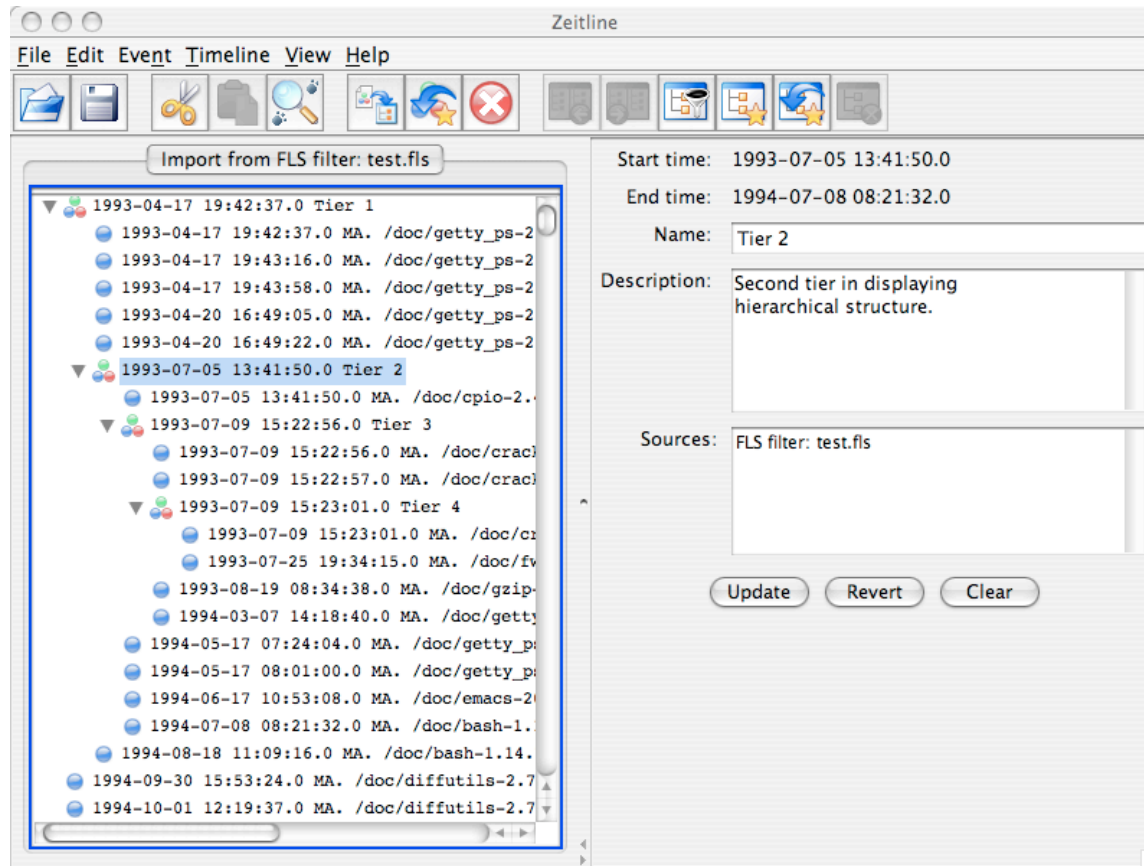
User Interface



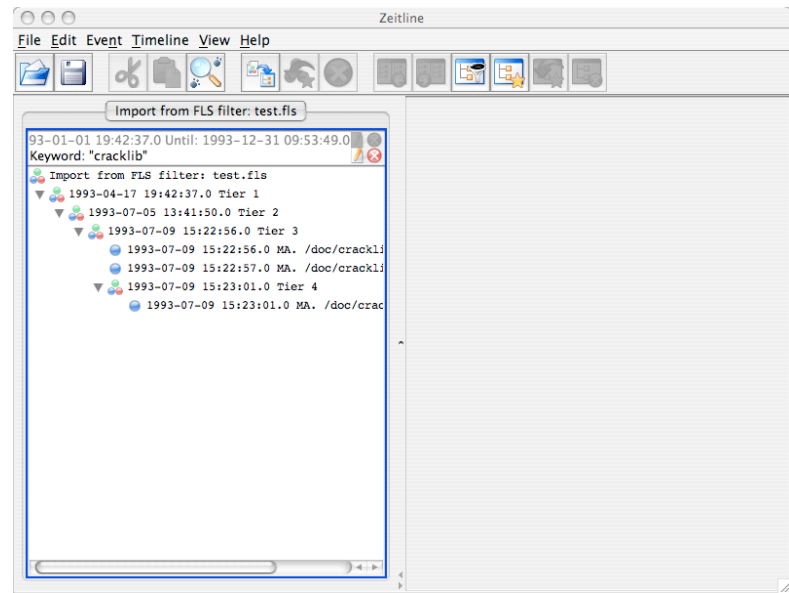
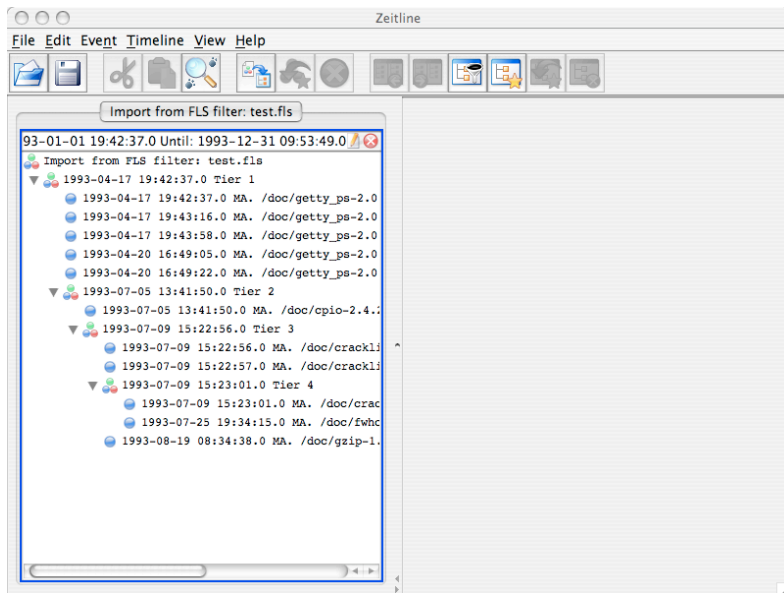
Importing Events



Hierarchical Structure



Queries



Dynamic Filters

- All input filters, including those provided with the package, are dynamically loaded
- Overwhelming task of writing filters for all possible input sources
- Writing new filters is as easy as implementing a single abstract class and doesn't require recompiling the entire project

InputFilter Class

```
public abstract class InputFilter
{
    public abstract Source init(String location,
                               Component parent);
    public abstract AtomicEvent getNextEvent();
    public abstract FileFilter getFileFilter();
    public abstract String getName();
    public abstract String getDescription();
    public abstract long getExactCount();
    public abstract long getTotalCount();
    public abstract long getProcessedCount();
}
```

Future Features

- Increased host information: user mappings, hardware and software details
- Events with no time value (logical events)
- Events with multiple hosts: clock synchronization, drift, and time zones
- Typing system for events
- Exportation such as XML output