# LogView: Visualizing Event Log Clusters

Adetokunbo Makanju, Stephen Brooks, A. Nur Zincir-Heywood, Evangelos E. Milios
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia
B3H 1W5
Canada
{makanju, sbrooks, zincir, eem}@cs.dal.ca

*Abstract*— **Event logs or log files form an essential part of any network management and administration setup. While log files are invaluable to a network administrator, the vast amount of data they sometimes contain can be overwhelming and can sometimes hinder rather than facilitate the tasks of a network administrator. For this reason several event clustering algorithms for log files have been proposed, one of which is the event clustering algorithm proposed by Risto Vaarandi, on which his Simple Log file Clustering Tool (SLCT) is based. The aim of this work is to develop a visualization tool that can be used to view log files based on the clusters produced by SLCT. The proposed visualization tool, which is called LogView, utilizes treemaps to visualize the hierarchical structure of the clusters produced by SLCT. Our results based on different application log files show that LogView can ease the summarization of vast amount of data contained in the log files. This in turn can help to speed up the analysis of event data in order to detect any security issues on a given application.**

## I. Introduction

An event log or log file consists of several independent lines of text data, which contain information that pertains to events that occur within a system. A log file might contain events from one service or different services which may come from one node or several nodes on the network. The actual setup is usually at the discretion of the administrator.

For this reason the contents of event logs are an important indication of the current status of the system(s) that they monitor. This makes them indispensable in systems administration and network management, are used by administrators in their general monitoring tasks, security analysis and also for trouble shooting when downtimes occur.

While log files are invaluable to a network administrator, the vast amount of data they sometimes contain can easily overwhelm a human and can actually hinder rather than facilitate the tasks of an administrator. A possible solution to this problem is to cluster the events based on their type. Data mining techniques are required to produce such clusters. One such technique is the frequent event mining algorithm [1], on which the Simple Log file clustering tool (SLCT) [2] is based. SLCT is a text based tool. The clusters produced by such methods can be used in tasks such as event correlation or in system profiling both of which can be used in fault detection and anomaly detection in a security context.

Despite the promise provided by data clustering tools such as SLCT in providing useful insight into the character of a log file and invariably the specification of the "normal" behavior of a system, they have not received much attention in network and system management tools in practice. Moreover, visualization techniques can help interpret large amounts of data, and log file analysis could definitely benefit from visualization as well.

The aim of this work is to build a visualization tool that can be used to view event log clusters such as the ones produced by SLCT. A major cornerstone of our work is to not just build a visualization tool but to build one which is interactive, dynamic and based on arbitrary clusters of application event/log files. Several tools have been proposed in recent times for the visualization of network data e.g. Afterglow [3], SnortView [4], VISUAL (Visual Information Security Utility for Administration Live) [5], Session Viewer [6] and HNMaps [7]. So far none of these tools provides all the cornerstones of our project in one package.

Our tool will provide visual summaries of the contents of an event log file to an administrator, to speed up the data analysis needed during downtimes and security breaches. The tool, which is christened LogView, utilizes treemaps [8] to visualize the hierarchical structure of the clusters produced by SLCT. Treemaps are used as the visualization technique in this project as they provide an appropriate alternative to traditional node-link diagrams used for viewing hierarchical structures. Node-link diagrams do not use space efficiently. This fact also makes visualization tools like daVinci [9](now called uDraw) inappropriate for our work. daVinci is an extensible, multipurpose visualization tool for hierarchical node link diagrams. Treemaps are also useful when the intent of visualization goes beyond the need to visualize the structure of the hierarchy but also requires the encoding of other pieces of information.

The rest of this paper is organized as follows: section 2 provides an overview of SLCT and treemaps and describes previous work. Section 3 discusses the steps taken to achieve our goal i.e. going from data collection to data preparation and finally the visualization of the data. Section 4 describes the results whereas section 5 presents the conclusion and future work.

## II. Background and Previous Work

### A. Previous Work

Visualization literature is abounding with examples of tools intended for use in the network management and security domain. These tools are usually designed with the goal of summarizing system related data for ease of analysis and differ mainly based on the type/source of data they visualize, the visualization technique utilized and task for which they are intended. Examples of recent network information visualization tools include Afterglow [3], SnortView [4], VISUAL (Visual Information Security Utility for Administration Live) [5], Session Viewer [6] and HNMaps [7].

Afterglow is an example of an open source toolkit/application which was designed for network information visualization. Afterglow consists of two modules 1.x and 2.0. Afterglow 1.x is a toolkit; it consists of a collection of scripts written in Perl which can be used to transform the contents of log files into a form which is suitable for generating visualizations. Afterglow 2.0 on the other hand is a treemap visualization application and takes files produced by Afterglow 1.x as input. Afterglow 2.0 is written in Java, and unlike our proposal, is designed with only the capability to view log files based on groupings of source IP addresses and intrusion detection log files, but not application log files.

SnortView was designed specifically for the visualization of Snort [10] alert logs. VISUAL visualizes communication patterns between hosts on an internal network and external hosts, while Session Viewer provides a visual alternative to the statistical approaches for the analysis of web session logs.

HNMaps are another treemap based information visualization tool proposed for visualizing internet level network traffic based on IP address hierarchies. HNMaps were applied to depict a network which spanned 7 continents, 190 countries, 23054 autonomous systems and 197427 IP prefixes [7].

Unlike the visualization tools discussed above, the proposed system is the only one which is designed with the visualization of event logs in mind. The proposed system also goes one step further by providing not just a visualization of raw data but data which has been preprocessed through an initial clustering phase. Moreover another contribution, of our work is to build a tool that is both dynamic and interactive. It should be noted here that to the best of our knowledge none of the above work has addressed these three issues together. Table I provides a comparative summary of the tools previously highlighted and our proposed tool.

### B. Simple Log File Clustering Tool

Several data mining algorithms for finding clusters in large databases with high dimensionality have been developed over the last decade. Examples of these clustering algorithms include CLIQUE [11], CURE [12] and MAFIA [13]. Unfortunately, though event logs (which are the primary source of data in network and system management) fit these criteria, these algorithms are in some respects not well suited for event logs. An algorithm suitable for clustering event logs needs to not just be able to deal with high dimensional data, it also needs to be able to deal with data with different attribute types, ignore the order of the input records and discover clusters that exist in subspaces of the high dimensional data [1], [14]. For this reason clustering algorithms which are designed specifically with event logs in mind are required, examples of such algorithms include simple event log clustering algorithm [1] and CUFRES [14]. The former algorithm deals with mining frequent patterns from event logs while the latter deals with clustering of events for correlation. The event log clustering tool, SLCT [2] (which stands for Simple Log File Clustering Tool), utilized in our work is based on the simple event log clustering algorithm. The algorithm and the tool were both proposed by Risto Vaarandi.

SLCT is an appropriate clustering tool for our work due its simplicity and its production of results which are comprehendible by humans. Aside from this, SLCT has been used successfully in intrusion detection[15] for the automatic creation of new attack detection signatures from log files.

Also the application event logs utilized in our work follow the typical pattern of word frequency occurrence in log files i.e. the majority of words occur infrequently, sometimes no more than once and a strong correlation exists between words that do occur frequently, due to the fact that every line in an event log is formatted according to some format string. The constant words in the format string therefore occur frequently in log files. These peculiar properties of event logs were utilized in the design of SLCT [1], which also another reason why SLCT is well suited to our work. The technique used in SLCT works by using a 3-step algorithm, with the first step proceeding in a fashion that is very similar to the Apriori algorithm for association rule mining. During the first step frequent 1-item attributes are identified, these frequent 1-items are then used to build cluster candidates in the second step. In the third and final step clusters are selected from these candidates if the number of lines they match is greater than a threshold provided by the user.

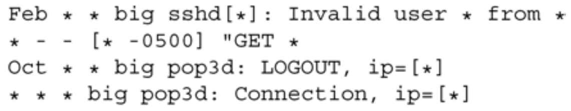
Fig. 1 shows four examples of the type of clusters that SLCT is able to find, the asterisks (or wildcards) in each line indicate place holders that can match any word.

Any line that matches any of these patterns is a member of that cluster.

TABLE I

Comparing LogView to other Network Information Visualization Tools

| Name | Technique | Data Source | Purpose |
|---|---|---|---|
| Afterglow | Treemaps | Traffic Logs | Security |
| SnortView | 2-D plots | Intrusion detection logs | Security |
| VISUAL | Fan-in/Fan-out layout | Packet Trace Data | Security |
| Session Viewer | Various | Web Session Logs | General |
| HNMaps | Treemaps | AS Level internet traffic logs | Security |
| LogView | Treemaps | Application Logs | General |



```
Feb * * big sshd[*]: Invalid user * from *
* - - [* -0500] "GET *
Oct * * big pop3d: LOGOUT, ip=[*]
* * * big pop3d: Connection, ip=[*]
```

Fig. 1. Sample clusters generated by SLCT

SLCT is written in C and can be easily compiled using gcc. It can be downloaded from [2]. Some of the more important parameters required to run SLCT include:

- **Support Threshold (-s):** This can be given as a percentage or integer. Since SLCT determines clusters by identifying line patterns that occur frequently, the support threshold value is used to determine frequency. A line pattern in any log file needs to have members that are greater than or equal to the support threshold to be considered frequent. The support threshold can also be used to increase or decrease the number of clusters formed.
- **Byte Offset (-b):** This can be used to filter out irrelevant information that usually occurs at the beginning of log file entries e.g. timestamps. The value is in bytes and when specified SLCT ignores this number of bytes at the start of each line.
- **Outliers (-o):** Specifies a file where outliers are stored i.e. lines that do not fit into any clusters.
- **Ignore (-f <regexp>):** This option takes a regular expression as input and tells SLCT to ignore any log entry that matches the expression when processing the log file.

### C. Treemaps

Treemaps were first proposed by Professor Ben Shneiderman of the University of Maryland in 1992 [8]. His aim was to produce a visualization of the directory structure of the file system of his 80MB hard disk with a technique that utilizes a space-constrained layout. Treemaps provide an alternative to the node-link structure diagrams traditionally used for visualizing hierarchical data. Treemaps are particularly useful when visualizing large amounts of hierarchical data, as they allow data to be viewed in a confined space. They also provide an interface which can be useful for encoding other pieces of information, a convenience which is not readily available or convenient with node-link representations. This makes treemaps an excellent choice for visualizing log file event clusters.

An example of how a treemap can be used to represent the hierarchical structure of a node-link diagram is provided in Fig. 2. Leaf nodes are represented by numbers in Fig. 2 (b) while internal nodes are represented with letters. The number representation of each leaf node is also an indication of the size of the node; we can immediately notice that these numbers could have been omitted in the treemap representation as the size of each node is encoded by the size of its corresponding block in the treemap.

A treemap is usually produced using the classic slice-and-dice treemap layout algorithm. However, other treemap layout algorithms have been proposed [16] to solve the problems of low-aspect ratio, layout instability, order preservation in the face of dynamically changing data and the need to create layouts that are easy to search visually. Some of these layouts include; Squarified, Strip, Cluster, Pivot-by-spilt, Pivot-by-size and Pivot-by-Middle. Our tool utilizes the squarified treemap layout algorithm.

## III. Methodology

In this work the major steps taken can be summarized as follows:

- Data Collection
- Data Processing, Preparation and Clustering
- Cluster Visualization

The following sections explain these steps in more detail.

### A. Data Collection

To build and test a visualization tool like LogView, appropriate data is required. The data used in this project was collected on a test server from our Netpal project [17].

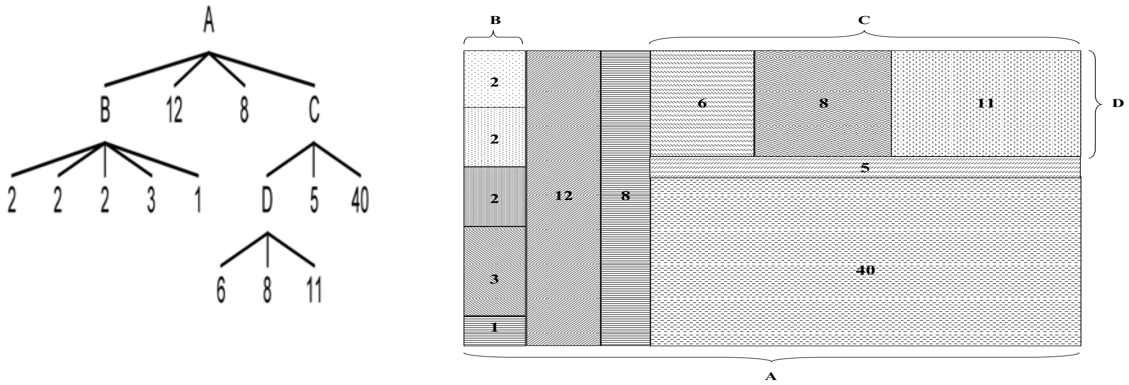
The log files were collected on a per service basis and log files from four services were used for this project. The services chosen include IMAP, POP3, SSH and HTTP. These services are explained in more detail in the following. The contents of the event logs employed in this work are summarized in Table II .

- **IMAP:** The IMAP service is an application layer protocol which allows a client to access his/her e-mails stored on a remote server over a TCP/IP connection. IMAP stands for Internet Message Access Protocol.
- **POP3:** Like IMAP, POP3 is also an application layer protocol which allows a client to access his/her e-mails on a remote server over a TCP/IP connection. It however differs from IMAP in that it does not allow persistent connections. This means that connections last for only as long as it takes to download messages, while POP3 allows clients to stay connected for as long as they require. POP3 stands for Post Office Protocol v3.
- **SSH:** SSH is an application layer protocol which allows two computers to exchange information over a secure encrypted channel. The information exchanged or transferred can be shell commands or files. SSH stands for Secure Shell.
- **HTTP:** One of the most popular application layer protocols in use today, HTTP is the protocol used for communication over the World Wide Web (WWW) and internal Intranets. HTTP stands for Hyper Text Transfer Protocol.

TABLE II

Dataset Summary

| S/No | Service | No. of Events | Period Covered |
|---:|---:|---:|---:|
| 1 | httpd | 574,664 | > 6 months |
| 2 | imapd | 85,721 | > 6 months |
| 3 | pop3d | 86,190 | > 6 months |
| 4 | sshd | 146,092 | > 6 months |



(a) Tree Structure

(b) Treemap representation

Fig. 2. Treemap visualization of hierarchical data using the slice and dice algorithm.

These event log files only form a fraction of the possible services that could be visualized by LogView or clustered using SLCT. It is possible to cluster and visualize the log files of virtually all services. These four services were selected only as samples for the demonstration of the tool.

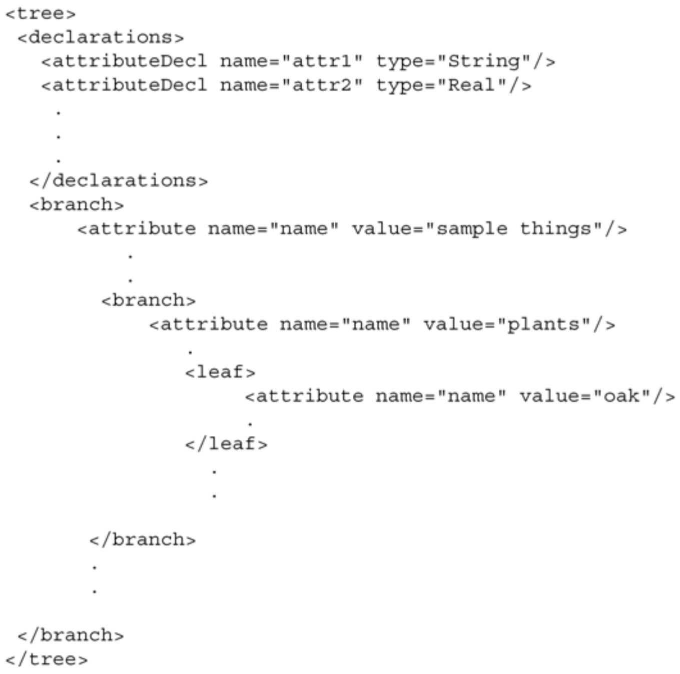
## B. Data Processing, Preparation and Clustering

Our visualization tool is not intended for the visualization of raw event log data but event log clusters. For this reason, the data collected had to be processed into a suitable form before it could be used as input to our visualization tool. The three steps required to process the data included running the data through SLCT to get the cluster representations, sorting of the events in the data set according to their clusters (as produced by SLCT)and producing TreeML files with the resulting data.

In the first phase of data processing and preparation, SLCT was run using the log files as input. After running SLCT, the raw clusters produced were inspected with the aim of selecting relevant and meaningful clusters, a task which is made easy by the human readable cluster representations produced by SLCT. A description of the clusters which were selected for visualization from each service are outlined in Table III. The name of each cluster was intuitively assigned.

In the second phase, regular expressions, which matched the selected cluster representations were produced. These regular expressions were then used to produce scripts, which are able to sort the events in each of the log files based on the selected clusters they belonged to; an event which did not belong to any cluster was classified as an “outlier”. This was necessary given the fact that SLCT’s output are not the clusters themselves but only textual representation of the form the events in each cluster should take, see Fig. 1. Since we have textual representations of each cluster, the regular expressions serve as input to the system for the automatic classification of log file events.

In the final phase, further scripts were written to take these event clusters and produce files suitable for input into our proposed tool. The TreeML format was chosen as the input file format. The TreeML format is an XML based file format designed for the purpose of specifying the data in a tree hierarchy. The general outline of a TreeML file is given in Fig. 3


```
<tree>
 <declarations>
   <attributeDecl name="attr1" type="String"/>
   <attributeDecl name="attr2" type="Real"/>
     .
     .
     .
  </declarations>
  <branch>
      <attribute name="name" value="sample things"/>
           .
           .
        <branch>
            <attribute name="name" value="plants"/>
                 .
                <leaf>
                     <attribute name="name" value="oak"/>
                     .
                </leaf>
                  .
                  .

         </branch>
         .
         .

  </branch>
</tree>
```

Fig. 3. General outline of a TreeML file

The TreeML format requires that we declare a number of data attributes which can be assigned to the entities in the tree. Table IV gives an outline of the data attributes used in the final TreeML files produced.

## C. Cluster Visualization

The visualization component was built using a combination of the Java Swing GUI toolkit and the prefuse visualization toolkit [18]. The prefuse toolkit was a natural choice for this implementation as it is implemented in Java and provides excellent classes for building visualization tools, which allow the user to interact with the visualization. Building an interactive visualization is one of the aims of this project.

The color coding scheme used in our tool is very simple. The outlines of the squares were drawn using a grayscale color coding scheme going from black through shades of gray to white, the color used for an outline gets progressively brighter, the deeper in the hierarchy that the user goes. When a leaf node is selected or a mouse pointer hovers over it, the outline of the node changes color to blue. Only the leaf nodes in the tree are colored. All leaf nodes are colored with different shades of green. The shade of green indicates the severity of the log event type with outliers having the darkest shade of green. The shade of green gets darker based on this severity order; OK, WARN, FAIL, OUTLIER. The color of the nodes however changes to red when the "msg" field of the node contains a search term entered by the user, more on this below.

LogView also offers a dynamic query, a search facility which allows text based searches of the log messages of the leaf nodes in the visualization. There is also a detail pane, which will show the "msg" field of a leaf node when it is selected and a drop down combo box, which switches the view between services. The dynamic query filters on the day of the month on which a log event occurred. The assumption here is that each input file will contain entries for one month only.

### *D. Prefuse Visualization Toolkit*

Visualization toolkits are becoming an increasingly popular means of creating information visualization tools as they help to reduce the time and effort required to build them. Some examples of such toolkits include Piccolo [19], InfoVis [20] and prefuse [18]. For the development of LogView, the prefuse toolkit is employed.

The prefuse visualization toolkit is a software framework written using the Java2D graphics library for the creation of visualization tools that are dynamic and interactive [21]. The design of the prefuse toolkit is based on the information visualization reference model outlined in the work of Chi [22]. By providing reusable building blocks based on this model for the easy building of custom visualization tools, prefuse goes beyond what is offered in other visualization toolkits.

Prefuse serves only as an application building toolkit in our work. It is transparent to the potential users of LogView.

## IV. Results

This section highlights the results of using the proposed system, LogView, to visualize the data collected. An overview of the LogView interface is shown in Figure 4, the example shows the visualization of the contents of the SSH service event log.

The components of the LogView window as can be seen in Fig. 4 include a service selection combo-box is at the top of the screen, a dynamic query slider to the middle-right and the visualization itself to the middle-left. There are also search and detail panes which occupy the bottom-right and bottom-left of the screen respectively. The labels for each cluster are the "names" given to each of the cluster types.

### *A. Service Overview and Profiling*

An overview of the visualizations created for each service type is shown in Fig. 5. Using such views, we can proceed to profile a particular service on a network. In this case, we can see that the IMAP and POP3 event logs are relatively less complex when compared to HTTP and SSH, since all their entries were able to fall into defined clusters without outliers. Specifically IMAP clusters are the least complex with only two clusters.

Looking also at the SSH visualization, we can see that the majority of the entries fall into one cluster, this cluster represents invalid login attempts. Further investigation showed that this was due to a prevalence of brute force login attempts on this server. In this case, LogView provided a very nice visualization of such an attack attempt on the SSH server. For the HTTP visualization, we notice that most of the clusters are PHP related. This is an immediate indication that this web server runs mostly pages developed using PHP. Such a scenario would not occur on a web server that does not host such pages.

### *B. Data Analysis and Interaction*

In demonstrating the data analysis and interaction capabilities of LogView, we split the possible tasks into three i.e. searching, filtering and selection.

A screenshot of a search over the SSH service showing log entries which contain the term "root" is shown in Fig. 6, (a) this is a task that an administrator might want to perform over this service to find the frequency of attempts to gain root access over SSH for instance. In figure 6 (b), we see the filtering of the same visualization using the dynamic query. The filter is set to show only those entries that occur on the 27th day of the month, all other nodes are invisible. The view can be dynamically changed by simply dragging the slider. The slider has values in the range 0 - 31. The range 1 - 31 for each of the possible days of a month and 0 (the default value) means "show all nodes".

There might be times when the administrator has focused on a node of interest and wants more information on the node. LogView allows for such situations; by hovering a mouse pointer over the node the actual log entry gets displayed in the detail pane below. An example of such a selection operation is shown in Fig. 6 (c). As an illustration of a situation where a selection operation would be useful, imagine finding out that your server has been flooded with brute force login attempts, a selection operation on a log entry would reveal the IP address of the source of the flood.

### *C. Zooming and Panning*

For user convenience LogView provides the capability to zoom and pan the visualization. A screenshot of LogView zoomed out on the visualization of the SSH service is shown

TABLE III
CLUSTER SUMMARY

| Service | Cluster Names | Description |
|---|---|---|
| **SSH** | Invalid User | Shell login request from a non registered user. |
| | Failed Reverse Mapping | Failed attempt to use getaddrinfo() to resolve a hostname. |
| | Spoof IP | A request from an IP address that may be spoofed. |
| | PAM Authentication Failure: Legal User | Pluggable Authentication Module (PAM) login failure from a registered user. |
| | PAM Authentication Failure: Illegal User | Pluggable Authentication Module (PAM) failure from an unregistered user. |
| | Failed keyboard-interactive/pam | Failed keyboard interactive or PAM authentication. |
| | No identification String | Shell login request without login information. |
| **IMAP** | Connection | A client connection to the IMAP server. |
| | Disconnection | A client disconnection from the IMAP server. |
| **POP3** | Logout | A client disconnection from the POP3 server. |
| | Connection | A client connection to the POP3 server. |
| | Login Failed | A failed login attempt. |
| | checkmailpasswd: Login Failed | A failed login attempt associated with checkmailpasswd. |
| | checkmailpasswd: Connection | A client connection to the POP3 server associated with checkmailpasswd. |
| | checkmailpasswd: Logout | A client disconnection from the POP3 server associated with checkmailpasswd. |
| **HTTP** | PHP Undefined Index: Fail | PHP error associated with an undefined array index that generates a failure message. |
| | PHP Undefined Index: Warn | PHP error associated with an undefined array index that generates a warning message. |
| | PHP Undefined Index: Ok | PHP error associated with an undefined array index that generates an information only message. |
| | PHP Undefined Offset: 0 | PHP error associated with an undefined numeric array index of 0. |
| | PHP Undefined Offset: 1 | PHP error associated with an undefined numeric array index of 1. |
| | GET Request 500 | HTTP get request with a status code of 500. |
| | GET Request 400 | HTTP get request with a status code of 400. |
| | PHP Division by Zero | PHP error caused by division by zero. |

TABLE IV
DATA ATTRIBUTES USED IN TREEML FILES

| Attribute Name | Description |
|---|---|
| **Name** | A string describing the node. |
| **Entries** | An integer which indicates the number of events which are part of the entire tree rooted at that node. |
| **Cluster** | A string which represents the cluster description produced by SLCT. |
| **Severity** | A string describing the severity category of the even type. The categories are OK, WARN and FAIL. |
| **Service** | A string which indicates the service which produced the log entry. |
| **Msg** | The actual log event entry string. |
| **Server** | The name or IP address of the server on which the data was collected. |

in Fig. 7 (a), while Fig. 7 (b) shows the tool zoomed in and panned to the left over the same visualization. We believe that such utilities could prove valuable during the analysis of a log file.

## V. CONCLUSION AND FUTURE WORK

In this work, we developed a visualization tool, which displays event log clusters based on clusters produced by SLCT. Our proposed event log visualization tool, LogView, is developed using the prefuse visualization toolkit and the Java Swing toolkit. Moreover, a squarified treemap layout is employed as its visualization technique.

We demonstrated the usefulness of the prototype in carrying out various log analysis tasks including profiling, selection, filtering and searching on event log data sets from the following services i.e. IMAP, POP3, SSH and HTTP.

Future work will include refining the tool to allow users to choose the color coding scheme, and to automate the file preparation process. We also intend to apply machine learning techniques for event correlation and diagnostics using the output of LogView.

## ACKNOWLEDGEMENTS

The author would also like to thank staff of Palomino System Innovations Inc., based in Toronto for their support in completing this work. Moreover, the authors gratefully acknowledge the support of NSERC, MITACS and CFI.

This work is conducted as part of the Dalhousie NIMS Lab at http://www.cs.dal.ca/projectx/.

## REFERENCES

[1] R. Vaarandi, “A Data Clustering Algorithm for Mining Patterns from Event Logs,” in *Proceedings of the 3rd IEEE Workshop on IP Operations and Management*. Kansas City, MO, USA.: IEEE Press, October 2003, pp. 119 – 126.

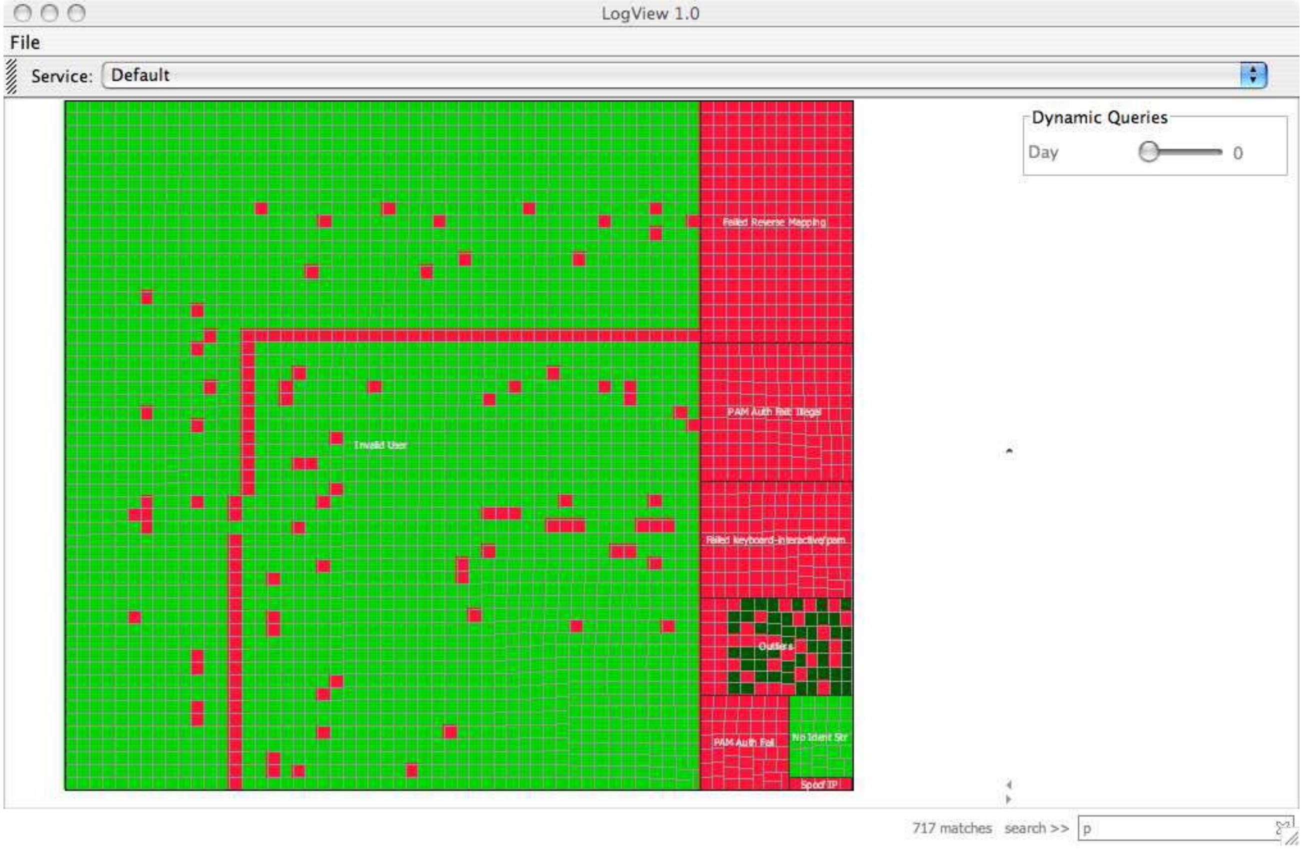
[2] ——, “Simple Logfile Clustering Tool : http://kodu.neti.ee/ risto/slct/,” Accessed from the web., 2003.

[3] R. Marty, “Afterglow Project Home: http://afterglow.sourceforge.net,” Accessed from the web., 2008.

[4] H. Kioke and K. Ohno, “SnortView: Visualization System of Snort Logs,” in *CCS Workshop on Visualization and Data Mining for Computer Security*. Washington, DC, USA.: ACM Press, October 2004, pp. 143 – 147.

[5] R. Ball, G. Fink, and C. North, “Home-centric visualization of network security traffic for security administration,” in *CCS Workshop on Visualization and Data Mining for Computer Security*. Washington, DC, USA: ACM Press, October 2004, pp. 55 – 64.

[6] H. Lam, D. Russell, D. Tang, and T. Munzner, “Session viewer: Visual exploratory analysis of web session logs,” in *IEEE Symposium on Visual Analytics Science and Technology*. IEEE Press, November 2007, pp. 147 – 154.



Service: Default

Day 0

717 matches

p

Fig. 4. An Overview of the LogView Interface.

[7] F. Mansmann, D. Keim, S. North, B. Rexroad, and D. Sheleheda, "Visual Analysis of Network Traffic for Resource Planning, Interactive Monitoring and Interpretation of Security Threats." *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1105 – 1112, December 2007.

[8] B. Schneiderman, "Tree Visualisation With Tree-maps: a 2-D space filling approach." *ACM Transactions on Graphics*, vol. II, no. 1, pp. 92–99, January 1992.

[9] M. Frohlich and M. Werner, "The graph visualization system da vinci - a user interface for applications," Department of Computer Sceince, University of Bremen, Germany., Tech. Rep., 1994.

[10] Sourcefire-Inc, "Snort - the de facto standard for intrusion detection/prevention, http://www.snort.org," Retrieved from the Web., September 2007.

[11] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," in *ACM SIGMOID International Conference on Management of Data*, 1998, pp. 94 – 105.

[12] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases." in *ACM SIGMOD International Conference on Management of Data*, June 1998, pp. 73–84.

[13] S. Goil, H. Nagesh, and A. Choudhary, "MAFIA: Efficient and Scalable Subspace Clustering for Very Large Data Sets," Northwestern University," Technical Report No. CPDC-TR-9906-010, 1999.

[14] J. Bellec and T. Kechadi, "CUFRES: Clustering Using Fuzzy Representative Events Selection for the Fault Recognition Problem in Telecommunication Networks," in *ACM International Conference on Information and Knowledge Management (Phd Workshop)*. ACM Press, November 2007, pp. 55 – 62.

[15] G. Hendry and S. Yang, "Intrusion signature creation via clustering anomalies," in *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2008. Edited by Dasarathy, Belur V. Proceedings of the SPIE, Volume 6973, pp. 69730C-69730C-12 (2008).*, March 2008.

[16] B. Bederson, B. Shneiderman, and M. Wattenberg, "Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies," *ACM Transactions on Graphics*, vol. 21, no. 4, pp. 833 – 854, October 2002.

[17] Precarn, "Netpal: Dynamic Network Admininstration, http://www.precarn.ca/products/projects/phase4/prjdusdcqkoyy.html," Accessed from the Web., May 2008.

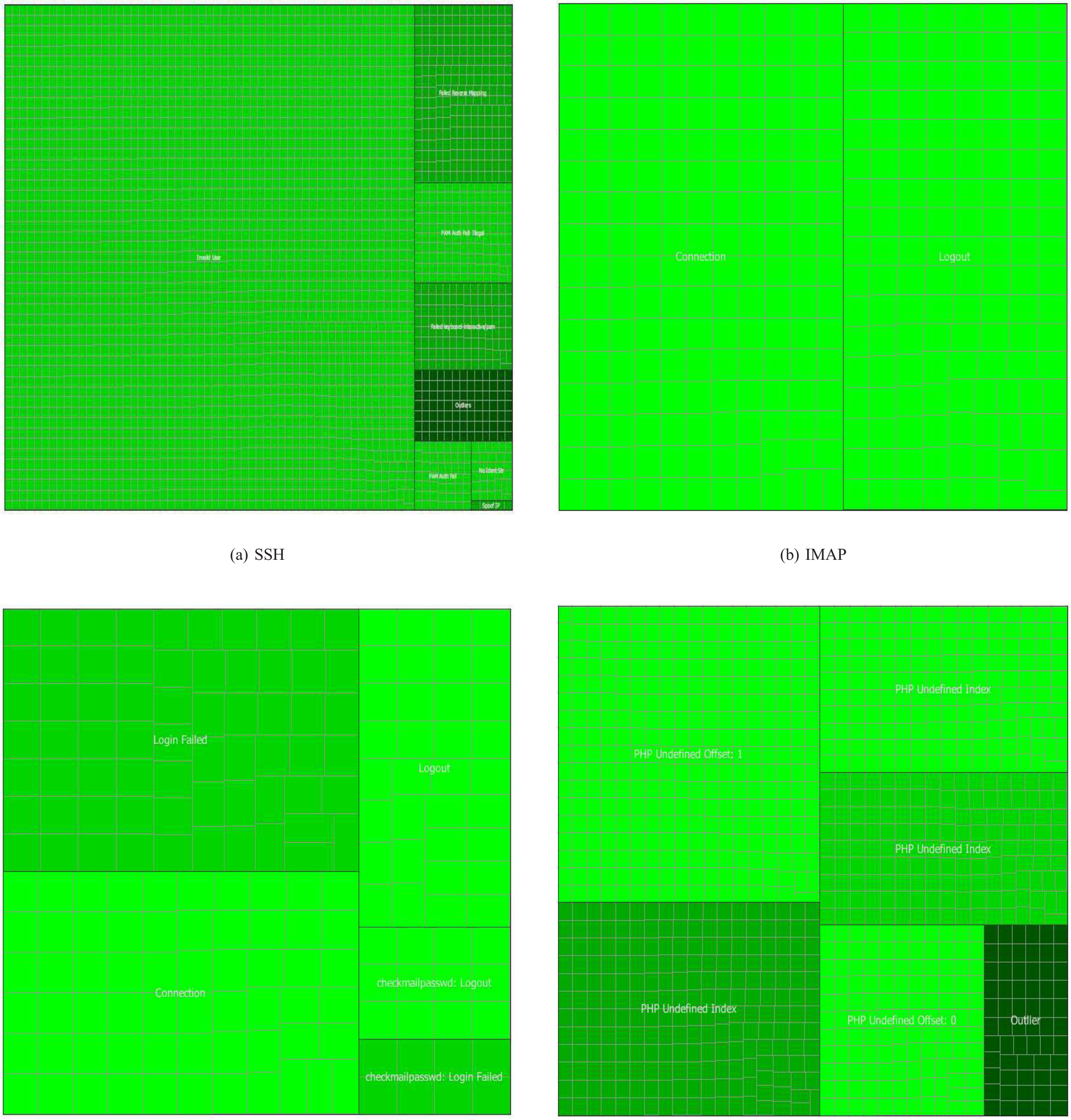
[18] J. Heer, "prefuse — interactive information visualisation toolkit, http://www.prefuse.org," February 2008.

[19] U. o. M. HCI Lab, "Piccolo Toolkit: A Structured 2D Graphics Framework, http://www.cs.umd.edu/hcil/piccolo/," Accessed from the web, March 2008.

[20] J. Fekete, "The Infovis Toolkit," in *InfoVis 2004*, 2004, pp. 167 – 174.

[21] J. Heer, S. Card, and J. Landay, "prefuse: a toolkit for interactive information visualisation," in *Proceedings of the Sigchi Conference on Human Factors in Computing*, New York, 2005, pp. 421 – 430.

[22] E. H. Chi, "A Taxonomy of Visualization Techniques Using the Data State Reference Model," in *InfoVis 2000*, 2000, pp. 69 – 75.



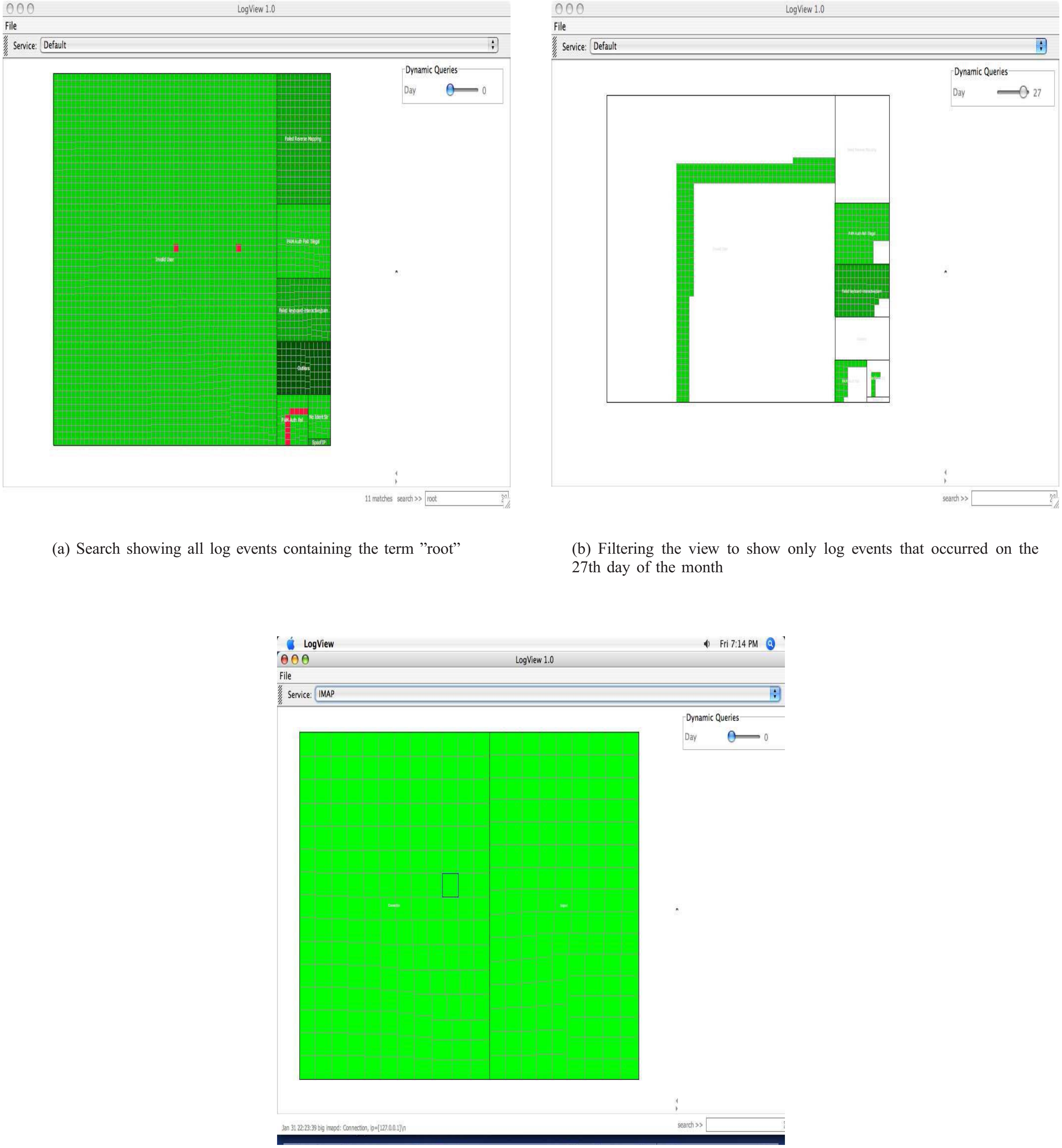
(a) SSH

(b) IMAP

(c) POP3

(d) HTTP

Fig. 5. Figure showing the Treemaps produced by LogView. (a)SSH (b) IMAP (c) POP3 (d)HTTP.



Service: Default

Day 0

11 matches search >> root

(a) Search showing all log events containing the term ”root”

Service: Default

Day 27

(b) Filtering the view to show only log events that occurred on the 27th day of the month

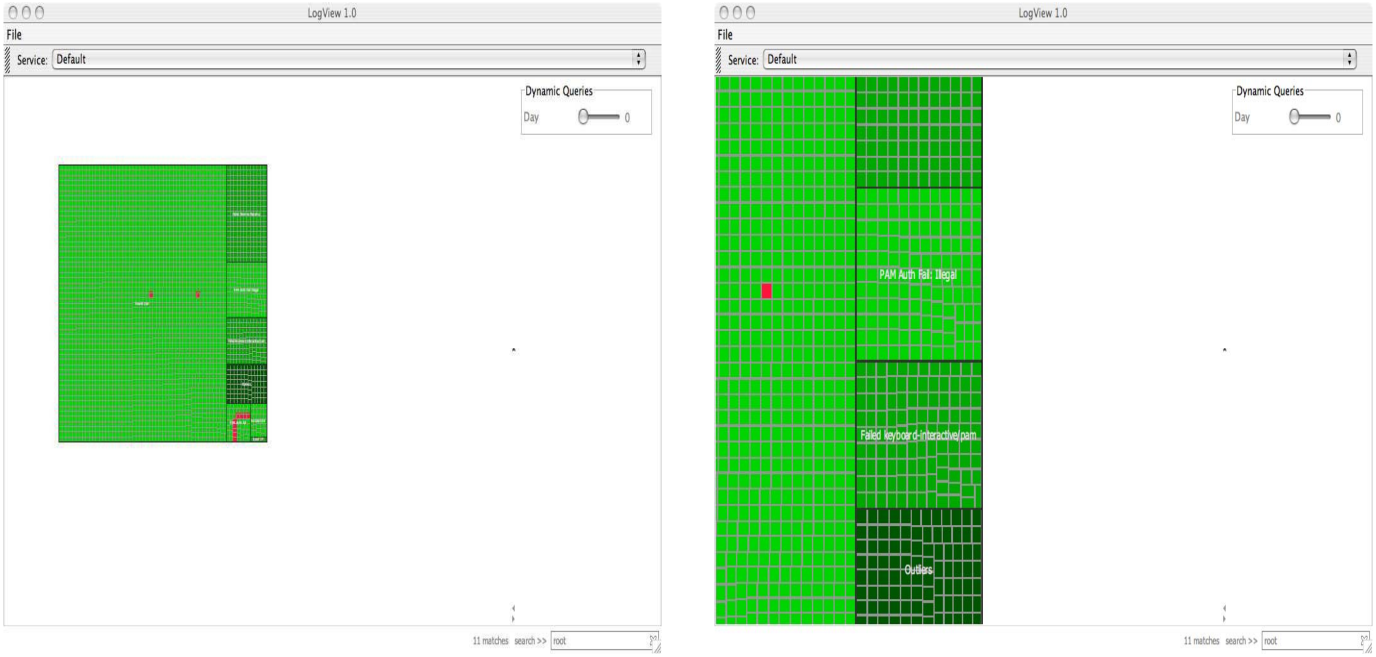
Service: IMAP

Day 0

```
Jan 31 22:23:39 big imapd: Connection, ip=[127.0.0.1]\n
```

(c) Selecting a node causes the message of the log entry to be displayed in the textbox below the treemap.

Fig. 6. Analysis Using LogView with the SSH service(a) Search (b) Filtering (c)Selection.



Service: Default

Day 0

search >> root

(a) ZoomOut

(b) ZoomIn and Pan-Left

Fig. 7. Zooming and Panning with the SSH service.(a) Zoomout (b) Zoomin and Pan-Left.