VisRAID: Visualizing Remote Access for Intrusion Detection

Leliel Trehothan[1], Craig Anslow[2], Stuart Marshall[1], and Ian Welch[1]

[1] School of Engineering and Computer Science
Victoria University of Wellington, New Zealand
leliel.trethowen@gmail.com, {stuart,ian}@ecs.vuw.ac.nz
[2] Department of Computer Science
University of Calgary, Canada
craig.anslow@ucalgary.ca

Abstract. Detecting malicious attempts to access computers is difficult with current security applications. Many current applications do not give the user the right information to find and analyze possible attempts. We present *VisRAID* – a novel visual analytics web application for detecting intrusions via remote access attempts, and a user study to evaluate the effectiveness and usability of the application with security professionals. The implications of the study will help inform the design of future security visualization applications.

1 Introduction

As computer networks have become ubiquitous threats to the integrity of the networks and data have multiplied and diversified extensively. The proliferation of threats has lead to the design of network applications to monitor for intrusions [14]. Intrusions are defined as access to systems resources for purposes contrary to the intended use of the system, or access to the system by an unauthorized person. Security policies exist as a formal statement of the intended uses of a system, and misuse can be considered malicious.

There are two main forms of malicious attacks that can be considered: the outsider and insider. The outsider does not have any legitimate access to the system and their attempts should be denied. The insider attack is significantly harder to detect as the attacker has legitimate access rights to the system, but is using them for unauthorized purposes. Insider attacks prevents simple solutions being implemented such as blanket denial of access. Further, this complicates detection of unauthorized access by obscuring illegitimate uses of the system within legitimate uses which should not be interfered with.

There are two main methods of operation for intrusion detection systems. Rule based systems flag any attempt that matches defined rules as malicious. These systems are extremely effective at detecting and blocking known attack patterns, particularly for outsider access as rules are written for the known attack patterns. Insider attacks are more difficult to control with rules based systems,

as blanket access blocks are often not suitable as it is not clear if a user is legitimate. Anomaly based systems use machine learning techniques to automatically classify incoming events as normal or anomalous. Normal events are ignored, while anomalous events are flagged for operator attention.

Anomaly based systems are able to recognize new intrusion methods but not able to provide much if any context about events, while Rule based systems identify which rule was matched. Anomaly based systems are harder to disguise existing intrusions from, as their classification systems are flexible enough to recognize small changes in patterns, whereas Rule based systems cannot do this as easily.

Intrusion detection system can suffer from false positives and false negatives. False negatives are undesirable as each represents an intrusion that went undetected and potentially un-countered. Note that not all undetected intrusions will achieve their goals, however security administrators are not able to effectively ensure the vulnerabilities used are addressed as they may remain unaware of the intrusion indefinitely unless traces are left elsewhere in the system.

Large numbers of false positives are problematic for secure systems as they quickly undermine the trust of users [11]. Many intrusion detection systems offer very limited forms of alerting with email mostly used, and SMS for critical alerts. The limited range of sensory urgency available via alerting mechanisms is problematic, as it causes a mismatch between the apparent and importance of a message [11].

Despite new intrusion techniques and rapidly growing networks, ensuring that actual intrusions are being detected is important. Existing intrusion detection systems have a number of problems: they do not scale well to large networks, cause information overload, and often fail to reveal important information about logged events.

In this paper we present *VisRAID* – a novel visual analytics web application that addresses these problems by allowing security administrators to effectively monitor remote access attempts to their systems to detect intrusions. VisRAID currently monitors SSH remote access attempts as the ports used for SSH traffic are often the focus of intrusion attempts. We present the design, user interface, and implementation of VisRAID, a preliminary user study with security professionals, and discuss implications for the design of security visualization applications.

2 Related Work

Related work on visualization of intrusion detection has primarily focused on two major areas: techniques used to visualize and explore the results of data mining, and visualization driven techniques. We treat intrusion detection systems (IDS) as datamining systems. Most of the research to date has focused on network intrusion detection logs, and relatively little work has considered access or system log visualization which is the primary focus of our research.

2.1 Data Mining with Visualization

LogView is a visualization application designed to support the understanding of information extracted through data-mining techniques applied to systems logs [9]. The visualization component uses treemaps to show clusters of events in a space efficient manner, with leaf nodes representing events, and branches showing clusters the events belong to. All leaf nodes are coloured in green, with shade darkening in four steps representing different statuses: OK, WARN, FAIL, and OUTLIER. The application allows filtering based on time, and search terms. Time filtering shows only events occurring on the specified day in the map. Nodes matching search terms are highlighted red. Detailed information about a given event is available via a mouseover. The simple filtering and interaction methods, coupled with very similar shades for clusters leaves the system very vulnerable to producing information overload in users. The overload potential grows rapidly as the logs grow, as there is very little information hiding present in this application.

Itoh et al. created a hierarchical system for visualizing intrusion system detection logs [7]. Intrusion detection system messages are lacking in the context needed to support an evaluation of the priority and accuracy of the report. The application shows the entire network under surveillance using a rectangle packing algorithm to group machines by subnets. The number of incidents sent and received from a given machine are displayed as a coloured bar rising out of the plane. The amount of data produced quickly leads to navigability and readability problems as the number of incidents reported grows. Simple filtering applications are available to limit by severity, time, IP, and signature ID. The application is highly reliant on an IDS with both a low false positive, and false negative rate, as almost all information about the actual events is hidden, and there is no easy method to drill down to detailed information.

Xu et al. present a system log data mining application [13]. The log syntax is automatically recovered through source code analysis, allowing the system to be applied to any source available. Once log syntax is extracted, logs are parsed and machine learning algorithms used to perform feature extraction. Once the features are extracted the principle component analysis anomaly detection algorithm are applied to this data, to find interesting patterns in log messages. This approach identified many issues in the tested software, but was forced to be extended from a pure data-mining approach as users found the black box nature of data-mining algorithms caused difficulty in understanding why the results were as they are. To assist with this, a very simple decision tree visualization was added. This visualization was created with the intention of showing the logic used by the data-mining systems to give context for decisions. As the processing required for feature extraction and anomaly detection is highly parallel this approach readily scales to millions of entries, and shows very strong performance in detecting anomalous patterns in logs. From a security standpoint, however, decision trees alone do not give sufficient context to easily determine if an anomaly represents an intrusion as the the trees depends on data not always included in the raw logs.

2.2 Visualization Driven Techniques

Picvis is an application created to generate parallel co-ordinate plots of log data [12]. Picviz has features to automate data extraction from several common log formats for use with their plot description language. This language allows a great deal of control over what variables are shown on a plot. Parallel co-ordinate plots can show strong clustering extremely well. However there are issues of scalability, as clusters and patterns can easily be hidden in background noise unless axes are well chosen. Filtering features are available to help address this limitation, but still require significant knowledge of the data structure and content. This application would be excellent for confirmatory analyses however.

Spiralview is an application that displays time-series data which uses a spiral visualization technique and has been used to visualize static logs and dynamic data streams [1]. The aim of Spiralview is to reveal repeating patterns in time series data from intrusion detection system reports and access logs. An evaluation of the spiral technique asked participants to identify regions of high and low activity in cellphone calling records [3]. Participants were also asked to estimate the time period of patterns in the cellphone data. Participants were presented with both a spiral view and linear timeline views of the data. The study claims that the spiral view approach will offer higher performance than a linear timeline due to the reinforcement of repeating patterns. The results showed that users were significantly faster, more accurate, and more satisfied with a linear timeline presentation for identifying regions of high and low activity. Users were significantly more accurate at identifying time periods for repeating patterns with the Spiralview.

The integrated visualization system is an IDS log based visualization application, focusing on attacks originating inside the monitored network [10]. The application provides a unified logical, geographic and temporal display of data, using three orthogonal planes. Multiple layouts of the three planes are available, with animated transitions. The timeline plane in the visualization shows events for the entire subnet. Individual IP's can be chosen by shifting the timeline plane along the IP plane. Filtering features are made available to control what kinds of events are shown. The application relies on colour to distinguish ports on the timeline frame, which is easily subject to visual overload, as the human eye is not able to reliably distinguish fine differences in colours. Vertical position of the line indicates the number of events. With poorly described filtering systems and reliance on colour coding to distinguish ports, this system is extremely vulnerable to producing information overload. This leads to missing important events. The lack of data hiding creates visual clutter which can easily mask important intrusions composed of a small number of events. The application appears interesting as an attempt to correlate attack information with machine location through GeoIP systems. Where the number of events is "low enough" lines are drawn from IP plane to physical location on the lower plane. The exact number of lines drawn is not clear.

3 VisRAID

VisRAID is a novel web application for network security professionals to visually explore monitoring of network traffic for intrusion detection. VisRAID adopts a visualization driven technique design. There are several goals we have used to help guide the design of VisRAID:

G1: Deploy visualizations over the web.
G2: Strong filtering and highlighting options.
G3: Show surrounding context for anomalous accesses.
G4: Support sharing of work and saving work in progress.
G5: GeoIP support to add context to login attempts.
G6: Allow the user control over which machine is monitored at any given time.
G7: Show network context for currently monitored machine.
G8: Support extensible log parsing.

3.1 User Interface and Timeline Visualization

Figure 1 shows the web-based user interface of VisRAID. The main features are a timeline overview, vertical scaling, colour-coding of event types, and mouseover tooltips. Within each timeline, blocks are vertically scaled proportional to the block with the largest number of events to help with information hiding so important data is not masked. In Figure 1 the first block consists of 5 events total, the second block is absent indicating no events occurred in that time, and the third block shows 25 events of three different classes.

As networks can become extremely busy by producing lots of activity the information can overwhelm an analyst's ability to comprehend and detect meaningful patterns. To address this problem we adopted data hiding techniques, used a time series based approach for displaying the data, and used time binning to aggregate entities. In our approach all entries in a short time period are displayed as a single entity, with icons indicating some simple features of the hidden data. Such features include superuser accesses, number of abnormal failed access attempts, number of abnormal access attempts, abnormal login locations for a user, abnormal login times for a user. Each time a bin can be zoomed in on, allowing the analyst to see greater detail within the bin. For busy systems and longer time periods there may be multiple levels of binning to aggregate sufficiently. This approach reduces the visual complexity, but allows access to detailed information on demand.

Flags for abnormal login time and location are the most complicated flags, as they require creating a profile of each user's access times over repeated access attempts. This complexity can produce false positives while a user is learning the application and may be tripped up by a legitimate change in their behaviour.

Timelines are required to have independent vertical scales as the number of events in a given time period may be highly variable. This is demonstrated in Figure 1 where the top timeline shows 25 events in the largest bin, and the third timeline shows in excess of 20K events. Without independent scaling for each

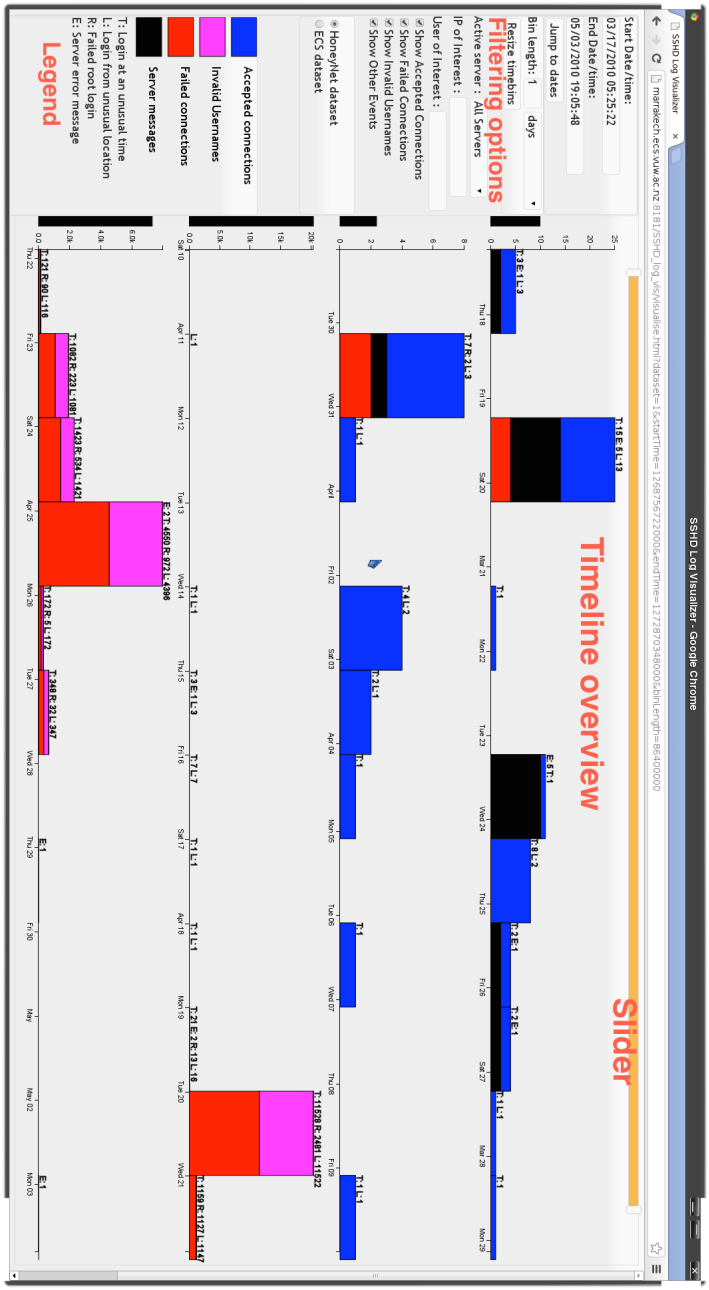

Fig. 1. VisRAID - showing an overview of the Honeynet dataset, slider to navigate, filtering options to make changes, and legend for different types of events.

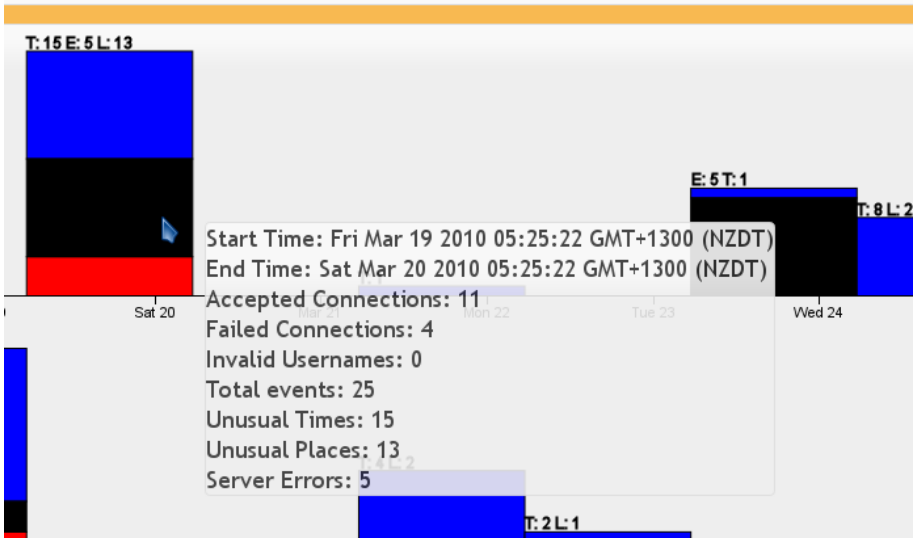

Fig. 2. VisRAID - Tooltip showing statistics for a block.

timeline all features of the upper two lines would be overwhelmed by the third. This led to the inclusion of the black scale indicators found between the controls and timelines. They are logarithmically scaled such that the timeline with the highest maximum events per bin is represented by a full bar. These provide a quick visual indication of how much variability there is between scales.

Each block is subdivided into four colours to represent different classes of events. Each event must be in exactly one event class, this allows the colour sections to be linearly scaled to block height (i.e. if half the events are failed connections, half of the block will be red). The colour coding was added to give an instant overview of the breakdown of event types within each bin. Hovering the mouse over a block produces a tooltip showing a detailed statistical breakdown of the events within that block (see Figure 2). Double-clicking on any block zooms in on that block, with all four timelines reloading to show only data from the selected block. Zooming can be performed until either there is only one event in the chosen block, or each block covers 1 second. Where there is only one event to display, mousing over the block will show the raw log line in a popup.

Above the timelines a slider can be used to navigate the dataset. The position of the slider gives an approximate indication of how far through the timeline the view is currently at. The extreme left is when the first log event occurred. The extreme right is the latest log event. The slider can be dragged and moved in steps. Each step covers exactly as long as is currently shown in the timelines (see Figure 3).

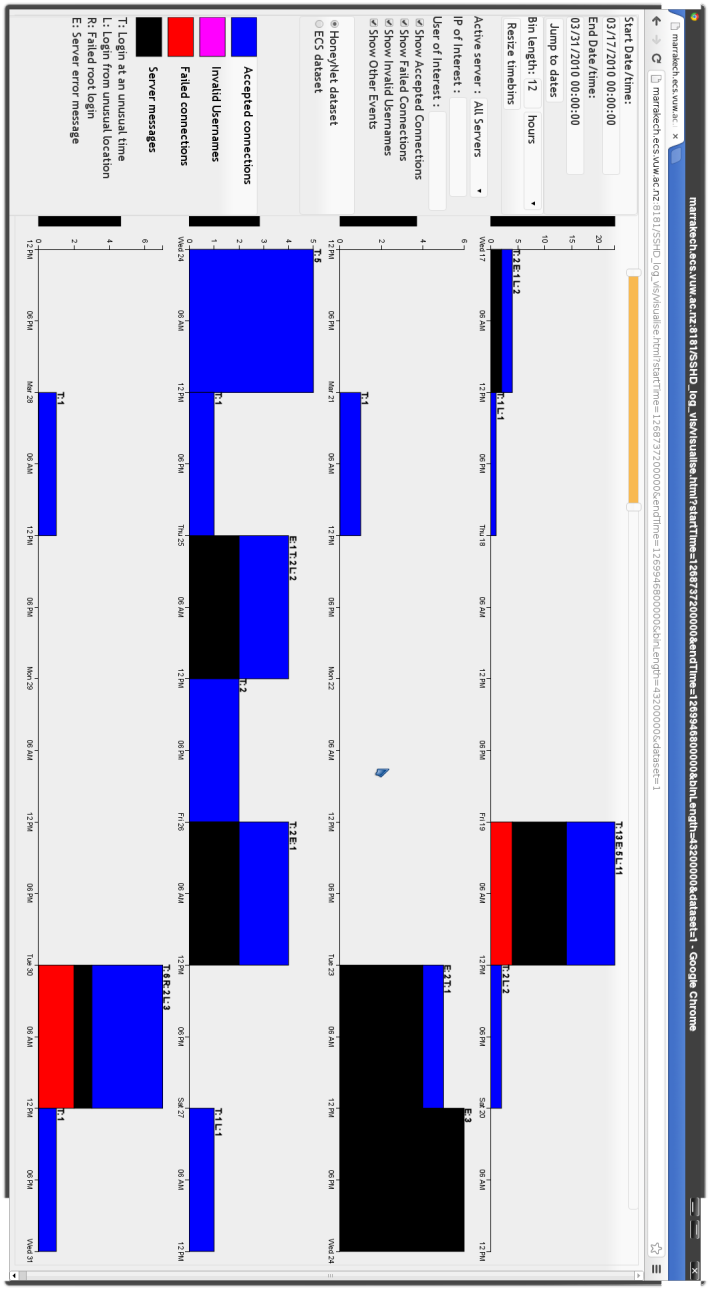

Fig. 3. VisRAID - Showing 2 weeks of the Honeynet dataset.

```
Mar 16 08:25:22 app-1 sshd[4884]: Server listening on :: port 22.
Mar 16 08:25:22 app-1 sshd[4884]: error: Bind to port 22 on 0.0.0.0 failed: Address already in use.
Apr 19 05:55:20 app-1 sshd[12996]: Accepted password for root from 219.150.161.20 port 55545 ssh2
Apr 19 05:55:20 app-1 sshd[12997]: Invalid user pauline from 219.150.161.20
Apr 19 05:55:21 app-1 sshd[12990]: Failed password for root from 219.150.161.20 port 54890 ssh2
Apr 20 00:00:51 app-1 sshd[24442]: subsystem request for sftp
Jun 8 01:03:34 machine0 sshd[1796]: Received disconnect from 38.165.101.19: 11: Bye Bye
```

Fig. 4. Examples of SSHD logs, showing each type of message.

3.2 Implementation

While there are many kinds of events, and metadata about connections that can be logged, extended information is highly dependent on SSH demon configuration. The listed message types can be relied on to be present in all useful logging levels. A log event will typically be represented by a single log line at the standard level of logging for OpenSSH. A typical user interaction may generate between two and three log events: connection, disconnection, and transferring files across the secure connection. Malicious access attempts may generate between two and three log events: connection, disconnection, and if the username provided is not recognised. Rates of log generation can vary widely between networks.

We used the following event types from the SSHD logs: connection attempts, disconnection messages, subsystem requests, invalid usernames, and system messages. All events contain a timestamp, server name, service name, and process ID. The remainder of a message is free text. Each entry contains metadata, for example a connection attempt contains information about authentication method, username, source address, and status. Whereas a disconnection message contains a code, and source IP. Figure 4 shows an example of some of the messages contained in the SSHD log files.

A log parsing tool was implemented in two layers: log reader and writer/analyser. The reader layer is responsible for reading any number of log files into a sorted list of events. The list of events is then passed to the analyzer/writer layer, which is responsible for checking connection attempts to see if the location and time are frequently used by the user and writing the analyzed metadata to the datastore. The results of the parser are stored in a MySQL database.

To address the design goal of deploying visualizations over the web we implemented a web-based application. The user interface was implemented with JQuery and visualizations with D3 [2]. Apache Tomcat was used for the server along with SQL generation from the jOOQ library [4]. The server contained two layers implemented as servlets. The data access servlet was responsible for fetching data from the underlying datastore. The servlet takes values from the client communication layer and returns lists of matching log entries. The servlet can be easily modified to communicate with different kinds of datastores. The client communication servlet was responsible for aggregating data returned by the data access layer, and building HTTP(S) responses from the aggregated data.

4 Evaluation

To evaluate the effectiveness of VisRAID we performed a preliminary user study with security professionals. We obtained human ethics approval from our University to conduct this user study.

4.1 Participants

Six people were recruited for the user study. Four of the participants were security professionals who had security components as part of their day jobs, and are regularly involved in log analysis tasks. The other two were computer science students who had passed a graduate security course.

4.2 Datasets

Two datasets were used in this study which represent real systems exposed to the public internet.

Honeynet Forensic Data: from Challenge 10 which is publicly available and anonymized [6]. The data covers a single server, with 35K log entries covering 16 March to 2 May (approx. 729 events/day).

ECS Data: from our computer science department which are network logs, anonymized, cover three servers for two disjoint weeks, and contain 74K log entries. (approx. 5300 events/day).

The Honeynet dataset has been extensively analyzed, over the course of two forensic challenges. Multiple successful brute force and scattergun attacks have been identified in this dataset. There were few usernames which showed a pattern of usage in the log data gathered. The Honeynet dataset was from the GMT-5 timezone.

The ECS dataset is more complex than the Honeynet dataset. The ECS dataset covers three separate servers, with more active users. The ECS dataset shows approximated 5300 events/day, more than seven times as many as the Honeynet dataset. The ECS dataset contains disconnection messages absent from the Honeynet dataset. These messages can account for at most a doubling of event rates, as there may be at most one disconnection per connection attempt, and a connection attempt may produce more than one event. As disconnection messages are not currently useful, they add a significant amount of noise. This dataset was extensively analyzed before the user study to search for existing attacks. Several attempted brute force and scattergun attacks were found, though all were unsuccessful.

The difference in attack success rates between datasets reflects differences in the purposes of the networks. The ECS network is provided for use by staff and students in our department at our university. This network can reasonably be expected to have significant amounts of sensitive information stored. Therefore the school expends significant effort in protecting these systems. In contrast, the

Honeynet system is deliberately exposed to lure in attackers, and as such has significantly less effort expended on securing it.

The ECS dataset was altered, to introduce a successful scattergun attack on one server. This was introduced, as there were no naturally occurring successful attacks discovered after thorough analysis with both VisRAID, and direct exploration through the database. The introduced scattergun attack in the ECS dataset had a relatively small attack signature, with only 204 log entries involved, of 4.7K entries for that day. Some successful attacks on Honeynet had similar numbers of access attempts, though often a much higher proportion of invalid or failed attempts for a given day.

The ECS dataset is recorded from a live network in normal operation. The log produced contains IP addresses and usernames of everyone to use the system during the recording period. As this information could easily be used to identify people, it requires being anonymized to be ethically used without each user's consent. SSHD logs contain usernames and raw IP addresses. Both usernames and IP addresses can be useful to malicious people. IP addresses allow for approximate location of a user's home through GeoIP databases, as well as direct attacks on the security of their computers. Usernames do not carry as significant a risk to the owner, though these could be useful to mount an attack on the ECS system directly.

Both datasets were anonymized, but the procedures were different. The Honeynet dataset is publicly available in an anonymized form, hence we did not have to do anything further to the data [6]. The ECS dataset was recorded from Internet facing servers in the ECS network. The ECS dataset was anonymized by system admin staff within our department. The data required anonymizing both usernames and IP addresses. IP addresses were anonymized with CryptoPAN a prefix preserving IP address anonymize tool extended with support for IPv6 addresses [5]. CryptoPAN uses strong encryption to generate codes which are combined with the IP address to produce a new valid IP address. This cannot be reversed without knowledge of the key, or an efficient means of cracking AES encryption. Usernames were anonymized through a script, where each username was replaced with the string "user" and a unique number.

4.3 Procedure

The user study was conducted in a controlled lab, with only the participant and session instructor present. Audio recordings were made as a record of events during the study as an addition to handwritten observational notes. Participants were provided with a desktop computer running at 1920x1080 resolution and the Chrome web browser.

Participants were given up to ten minutes to familiarize themselves with VisRAID. After familiarization, participants were asked to answer four questions about each dataset for a total of eight tasks. Questions were presented to users in a random order to avoid any learning bias. Participants were given up to eight minutes for each task. The participants were not given the opportunity to read

questions before the study began which limited their opportunity to learn the answers to later questions while answering a question.

The four questions are listed below. Tasks 1 through 4 were based on questions 1 and 2, with tasks 1 and 4 using the ECS dataset, and tasks 2 and 3 using the Honeynet dataset. Tasks 5 through 8 were based on questions 3 and 4, with tasks 5 and 7 using the ECS dataset, and tasks 6 and 8 using the Honeynet dataset. Table 1 lists the combinations of tasks, questions, and datasets.

Q1 Find an instance of a successful brute force attack on root.
Q2 Find an instance of a successful scattergun attack. (an instance where the attacker attempts many common username/password pairs at random).
Q3 Find an instance of a legitimate user logging in from an abnormal location.
Q4 Find an instance of a legitimate user logging in at an abnormal time.

Q1 and Q2 are based on the most commonly found attack signatures in SSH logs with many botnets and automated systems carrying out brute force or scattergun attacks against any IP address responding to connection requests. As these attacks are very common and can lead to serious compromises determining success or failure of such attacks is a core function of any log analysis tool. Q3 and Q4 are based on finding anomalous behaviour by legitimate accounts. Anomalous behaviour by legitimate accounts can be an indication that their account has been compromised, or that the account owner has become malicious.

For each task a brief questionnaire was completed indicating participants' subjective opinions about different aspects of VisRAID [8]. Participants were asked to complete the following three questions using a 7 point likert scale, where 7 is Strongly Disagree, and 1 Strongly Agree.

1. I am satisfied with the *ease* of completing this task.
2. I am satisfied with the amount of *time* it took to complete this task.
3. I am satisfied with the *support* information (e.g. online help, documentation) when completing this task.

Timing and accuracy for each task was recorded. For each task a date, time, source IP, and where applicable username involved were recorded. This information combined with the dataset provides sufficient details to allow checking of answers for accuracy from the raw logs, database directly, or using VisRAID.

4.4 Results

Each Task was given a pass/fail grade based on accuracy. Time taken to complete each task was also measured. Results are shown in Table 1. A dash indicates the task was not completed in time, hence an incorrect answer. Ticks represents correct answers and crosses incorrect answers.

When participants were working on Tasks 1 and 4 (ECS), they had difficulties in navigating the timeline which was a significant aspect for carrying out these tasks. Task 1 involved finding a brute force attack which compromised root. There was no such attack present in the dataset. Demonstrating the absence

Table 1. Time and accuracy results for each task by participant (Professionals and Students). Dashes represent tasks not completed within 8 mins. Ticks represents correct answers and Crosses incorrect answers.

T#	Q	Dataset	P1- P		P2 - P		P3 - S		P4 - S		P5 - P		P6 - P		Correct
1	Q1	ECS	3:05	✓	5:46	✗	-	✗	-	✗	-	✗	-	✗	1
2	Q1	Honeynet	-	✗	-	✗	2:26	✓	2:09	✓	3:55	✓	3:08	✓	4
3	Q2	Honeynet	5:10	✓	-	✗	4:18	✓	4:51	✓	8:00	✓	2:20	✗	4
4	Q2	ECS	-	✗	-	✗	4:18	✓	3:18	✗	-	✗	-	✗	1
5	Q3	ECS	1:09	✓	1:22	✓	1:07	✓	0:39	✓	1:04	✓	1:10	✓	6
6	Q3	Honeynet	0:31	✓	1:07	✓	1:02	✓	1:13	✗	1:00	✓	2:15	✓	5
7	Q4	ECS	0:45	✓	1:59	✓	0:42	✗	0:44	✓	2:06	✓	-	✗	4
8	Q4	Honeynet	0:32	✓	1:10	✗	1:32	✗	1:07	✓	1:15	✓	0:55	✓	4
		Total	26.32	6/8	34.44	3/8	22.45	5/8	20.81	5/8	32.8	6/8	33.08	4/8	29/48

of an item in a dataset can be significantly harder than finding the presence of it similar to finding bugs in software. Only one participant was successful in completing Task 1 and we believe the combination of navigation difficulties in the visualization with increased task difficulty was the cause of the very high failure rate for this task. Task 4 involved participants looking for a successful scattergun attack in the ECS dataset. Only one participant was successful in completing Task 4. Poor navigation support caused problems as the relatively small attack signature was easily swamped in other data.

Tasks 2 and 3 (Honeynet) were to find a brute force and scattergun attack respectively. There were multiple successful brute force attacks, and scattergun attacks with much larger attack signatures (higher number of attempts). These questions were quickly and reliably answered by most participants. Poor accuracy, and significantly slower times with the ECS data coupled with observations of participants attempting these tasks suggest that navigation difficulties and limited filtering options were a greater issue in the more complicated dataset, with smaller attack signatures, and greater noise. Participants performed much better for tasks 5–8.

The introduced scattergun attack in the ECS dataset had a relatively small attack signature, with only 204 log entries involved, of 4.7K entries for that day. Some successful attacks on the Honeynet dataset had similar numbers of access attempts, though often a much higher proportion of invalid or failed attempts for a given day. There were many more legitimate access attempts to the ECS network, and much higher event density. Logging of disconnect messages introduced further noise to the system.

Participant 2 (Professional) had a great deal of difficulty in identifying brute force and scattergun attacks on both datasets. Feedback and observation of the participant in action suggest severe difficulties with navigation, combined with the lack of ability to hide all attempts from a specified set of IP addresses caused significant difficulties for this participant. Participant 2 commented that in the normal course of investigating such incidents using tools such as grep, they would build up a blacklist of IP's to hide from results as they were fully

investigated and discarded. VisRAID does not currently support this analysis feature. Participant 2 has significantly more experience in analyzing SSHD logs using traditional tools where stronger filtering tools are available, such as regular expressions.

Figure 5 shows the perceived effectiveness for the questions in the survey participants completed at the end of each task. Each question in the survey was answered with a score on a 7 point likert scale, where 1 is Strongly Agree and 7 is Strongly Disagree. There were variations between participants as could be expected in an early prototype, which can be caused by many factors. The results of the survey match well with the accuracy and time results, as each participant gave a higher score (Disagree) for tasks they found difficult. For ease and amount of time to complete a task the perceived effectiveness had a similar range with median of 2 and an outlier at 7. Amount of time did not have any outliers. For support the perceived effectiveness had a smaller range with a median of 3. The results shows that participants felt VisRAID was easy to use, allowed them to find the answers within a reasonable amount of time, but better support was required for helping participants to use VisRAID.

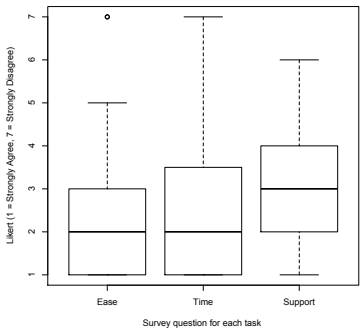

Fig. 5. Perceived Effectiveness of Ease of use, completion Time, and Support available.

5 Discussion

Based on our evaluation we discuss how VisRAID meets the design goals, weaknesses, suggested improvements, and limitations of the user study.

5.1 Design Goals

Several goals were presented to help guide the design of VisRAID. Some design goals were met while others were not.

G1: Deploy visualizations over the web. VisRAID was developed as a web-based application and can display visualizations using JQuery and D3 inside a web browser.

G2: Strong filtering and highlighting options. Participants were able to successfully answer questions about both datasets with VisRAID. Highlighting is not currently implemented. Extensions could be added to make the filtering stronger and implementing highlighting.

G3: Show surrounding context for anomalous accesses. The timeline display shows the surrounding time, with each level of zoom reducing the surrounding time that is visible. This provides context to users, but requires improvement due to navigation difficulties when transitioning.

G4: Support sharing of work and saving work in progress. These goals were not tested, but VisRAID is designed to support sharing of work through URL passing and saving work in progress through browser bookmarking.

G5: GeoIP support to add context to login attempts This is currently used for abnormal location detection, but is not currently made available to the user directly.

G6: User control over machine monitoring. Users have direct control over which machine's log data is shown at any time through the server menu, which is populated with a list of all servers known to the database.

G7: Show network context for currently monitored machine. Due to time this design goal was not met.

G8: Extensible log parsing. Only SSHD logs are supported, but integration of other syslog formats is possible.

5.2 Weaknesses

Navigation. Difficulties were experienced by most participants, where abrupt transitions between zoom levels lead to loss of context, and difficulty building up a mental map of the timeline. Most users demonstrated improved navigation as they became more familiar with VisRAID. These issues could be addressed in two major ways: showing a radar view to assist users in maintaining context, and animating transitions to build up a mental map of the log.

Filtering. Analysis of the difficulties experienced by participant 2, and suggestions from other participants several new filtering options would significantly enhance the ability to deal with potential information overload. Implementing IP blacklisting would be extremely useful for some users, as it would support an interaction model where one IP is fully investigated, then hidden, and the process repeated until all suspicious IP's have been investigated. One participant suggested allowing filtering by an authentication method. This is strongly supported by the difficulties users had in Task 1, as on multiple occasions, a successful root login would occur mixed in with many failed attempts. This successful login would be from a different IP address, using an authentication method not amenable to brute force, such as host-based authentication. Allowing filtering by authentication method, would assist users in avoiding this pitfall, by hiding a class of logins which cannot be involved in attacks.

Information Hiding. Information hiding is an issue for VisRAID as the timezone display caused all participants to query the discrepancy in times for the Honeynet dataset. This caused some confusion at first, however, results suggest that once informed of the issue users could compensate.

5.3 Suggested Improvements

Animated Zooming. Users complained that they get lost when zooming. A common feature used to smooth out such transitions and ease navigation is animating the zoom level. Animated zooming offers the potential for improvements by helping to address the navigation issues experienced by participants. As each timeline is currently updated independently and zooming replaces the contents of all timelines. The most common zooming action would be for the selected block to grow to fill all four timelines, replacing their data with a detailed breakdown.

Filtering by subnet. Adding options to filter IP addresses by subnet would potentially be useful, as it would allow more controllable filtering on IP than is currently present. Currently filters are restricted to matching against dotted quad forms of address. There would be some difficulty in implementing the filtering approach, as IP addresses are currently stored in human readable formats, not suitable for matching against less common subnets. Matching against subnets in 1 byte increments (/8, /16, /24) would be easy, as these match with the dotted quad format used.

IP Address Hiding. The inverse of the IP of interest filter is IP address hiding which shows all addresses except those selected. This would allow support of another interaction model, as used by participant 2. Implementing IP hiding, or blacklisting is relatively simple from a server side perspective, as the datastore is able to efficiently handle complex selection criteria. The difficulty is maintaining the stateful URL for this filter. Blacklists can grow potentially quite large, and URLs have an implementation defined maximum size. Larger URLs can be too long to email, hence a new approach may be needed.

Filter events by authentication type. This will help reduce false detection rates for brute force and similar attacks, as some authentication methods are not vulnerable to these attacks. Adding filtering for authentication type would offer the ability to hide all events that cannot be involved in specific types of attacks. This would be straightforward to implement but requires changes in several places. The data access layer of the server would have to be modified with an optional where clause in the query, and the URL would have to be modified with another optional filtering clause.

Suppress abnormal time and place warnings for invalid and failed attempts. This will cut down the number of abnormalities reported, and help to reduce false positives and information overload. The ability to spot abnormal logins would be improved. Implementation would require a change to the server, to ignore time and location flags when producing the aggregated data.

Separate disconnection messages into their own type. This would allow hiding of disconnection messages in the same way failed and successful connections can be hidden, reducing noise in the dataset. Implementation would

require changes in the aggregation code to count disconnections separately, and draw a 5th category of event.

Scaling to larger datasets. VisRAID was only tested on two datasets, where ECS was the largest and contained up to 75K log entries. Testing on much larger datasets will determine how well VisRAID scales and performs.

Larger Evaluation. Once VisRAID has been improved a further evaluation should be conducted with a larger scope, statistically significant sample size, and larger and more varied datasets to address the weaknesses. A further improvement to the evaluation procedure would be the inclusion of a training dataset to allow participants to become familiar with the application.

5.4 Limitations

There were some limitations with the user study and the datasets. The user study was conducted in a controlled lab environment, for a set period of time, and an application that the participants were not familiar with. Only six participants were involved, two of whom were computer science students. Students are less ideal for this kind of study, as their experience and domain knowledge are more limited than those that have been working in the industry for some time. Obtaining security professionals for user studies is difficult as we found there were a limited number in the city where the study was conducted. We could obtain more participants by evaluating in different locations. Small, exploratory evaluations have advantages in cost and time required. With small numbers of participants, the evaluation can be conducted in a short time and non-viable approaches can be discarded before significant development effort is invested.

Both datasets are in the 10's of thousands of entries, We have not tested VisRAID with much larger networks such as over 100K log entries. The addition of a successful scattergun attack to the ECS dataset represents a potential weakness of this study, however, great care was taken to ensure that the inserted log entries matched the patterns found in other scattergun attacks on both datasets. Participants were using datasets that they were not familiar with. Testing on much larger datasets in the future will determine how VisRAID scales.

6 Conclusion

Detecting malicious attempts to access computers is difficult with current security applications. Many current applications do not give the user the right information to find and analyze possible attempts. In this paper we presented *VisRAID* – a novel visual analytics web application for detecting intrusions via remote access attempts, and conducted a preliminary user study to evaluate the effectiveness and usability of the application with security professionals.

The user study involved six participants four of whom were security professionals. Participants were able to effectively answer the questions in the user tasks using different sized data sets. Some questions proved to be more difficult than others. The results showed that participants felt VisRAID was easy to use,

allowed them to find the answers within a reasonable amount of time, but better support was required for helping users learn the application. VisRAID could be improved by allowing easier navigation of the visualizations, providing better support for filtering by IP, and the ability to hide information more effectively.

References

1. E. Bertini, P. Hertzog, and D. Lalanne. SpiralView: towards security policies assessment through visual correlation of network resources with evolution of alarms. In *Proc. of Conference on Visual Analytics Science and Technology (VAST)*, pages 139–146. IEEE, 2007.
2. M. Bostock, V. Ogievetsky, and J. Heer. D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.
3. G. Chin, M. Singhal, G. Nakamura, V. Gurumoorthi, and N. Freeman-Cadoret. Visual analysis of dynamic data streams. *Information Visualization*, 8(3):212–229, 2009.
4. Data Geekery. jOOQ: Get back in control of your SQL. `http://jooq.org`. Accessed on 30 October 2013.
5. J. Fan, J. Xu, M. Ammar, and S. Moon. Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. *Computer Networks*, 46(2):253–272, Oct. 2004.
6. Honeynet Project. Forensic Challenge 10 - Attack Visualization. `http://www.honeynet.org/challenges/attack_visualization_challenge`. Accessed on 05 June 2013.
7. T. Itoh, H. Takakura, A. Sawada, and K. Koyamada. Hierarchical visualization of network intrusion detection data. *IEEE Computer Graphics Applications*, 26(2):40–47, 2006.
8. J. R. Lewis. IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 7(1):57–78, 1995.
9. A. Makanju, S. Brooks, A. Zincir-Heywood, and E. Milios. LogView: Visualizing Event Log Clusters. In *Proc. of Conference on Privacy, Security and Trust (PST)*, pages 99–108. IEEE, 2008.
10. S. Mukosaka and H. Koike. Integrated visualization system for monitoring security in large-scale local area network. In *Proc. of the Asia-Pacific Symposium on Information Visualisation (APVIS)*, pages 41–44. IEEE, 2007.
11. N. Stanton. *Human factors in alarm design*. CRC Press, 1994.
12. S. Tricaud. PicViz: finding a needle in a haystack. In *Proc. of the USENIX Conference on Analysis of System Logs*. USENIX Association, 2008.
13. W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan. Detecting large-scale system problems by mining console logs. In *Proc. of the Symposium on Operating Systems Principles (SIGOPS)*, pages 117–132. ACM, 2009.
14. Y. Zhang, Y. Xiao, M. Chen, J. Zhang, and H. Deng. A survey of security visualization for computer network logs. *Security and Communication Networks*, 5(4):404–421, 2012.