Text Mining for Security Threat Detection

Discovering Hidden Information in Unstructured Log Messages

Candace Suh-Lee
Dept. of Computer Science
University of Nevada Las Vegas
Las Vegas, NV, USA
suhlee@unlv.nevada.edu

Ju-Yeon Jo
Dept. of Computer Science
University of Nevada Las Vegas
Las Vegas, NV, USA
Juyeon.jo@unlv.edu

Yoohwan Kim
Dept. of Computer Science
University of Nevada Las Vegas
Las Vegas, NV, USA
yoohwan.kim@unlv.edu

***Abstract*—The exponential growth of unstructured messages generated by the computer systems and applications in modern computing environment poses a significant challenge in managing and using the information contained in the messages. Although these data contain a wealth of information that is useful for advanced threat detection, the sheer volume, variety, and complexity of data make it difficult to analyze them even by well-trained security analysts. While conventional Security Information and Event Management (SIEM) systems provide some capability to collect, correlate, and detect certain events from structured messages, their rule-based correlation and detection algorithms fall short in utilizing the information within the unstructured messages. Our study explores the possibility of utilizing the techniques for data mining, text classification, natural language processing, and machine learning to detect security threats by extracting relevant information from various unstructured log messages collected from distributed non-homogeneous systems. The extracted features are used to run a number of experiments on the Packet Clearing House SKAION 2006 IARPA Dataset, and their prediction capability is evaluated. In comparison with the base case without feature extraction, an average of 16.73% performance gain and 84% time reduction was achieved using extracted features only, and a 23.48% performance gain was attained using both unstructured free-text messages and extracted features. The results also show a strong potential for further increase in performance by increasing size of training datasets and extracting more features from the unstructured log messages.**

***Index Terms*—data mining, text classification, SIEM, Security Information and Event Management, unstructured log messages, natural language processing, named entity extraction, machine learning**

I. Introduction

A. Security Information and Event Management (SIEM)

The Security Information and Event Management (SIEM) system is a security system whose main function is to collect, store, search, and correlate system-generated log messages. System-generated log messages here refer to the messages generated by a machine in a human-readable format, in order to support maintenance, trouble-shooting, surveillance, or audit activities.

The three main functions of SIEM are: 1) aggregation of dispersed log messages in one central location for ease of access; 2) persistent storage of large amount of log data for a specified period of time; and 3) correlation of logs for threat or anomaly detection. The first two functions became important as the computer systems became more distributed and complex for the last a few decades. Also, regulatory or legal requirements of retaining relevant electronic records of important transactions justify the needs of SIEM. The third function stems from the idea of taking advantage of the centrality and normality of the log repository in order to gain better situational awareness of the systems. This type of bird-eye view provides a powerful tool in a security-sensitive context, where an adversary often "moves around" different devices attempting to gain unauthorized access in multiple different ways or compromises a series of devices in order to reach a target.

In order to support these functions, SIEM creates certain events by correlating multiple log entries from different systems according to the predefined correlation rules. If a certain event occurs, SIEM refers to the detection rules associated with the event to execute a relevant action, such as sending an alert email to the operator.

Figure 1 shows a typical SIEM architecture, where a number of remote log collectors or local log agents collect logs and send them to the central log repository. The log search and correlation engine provides search, correlation, detection, and alert services to the operator.

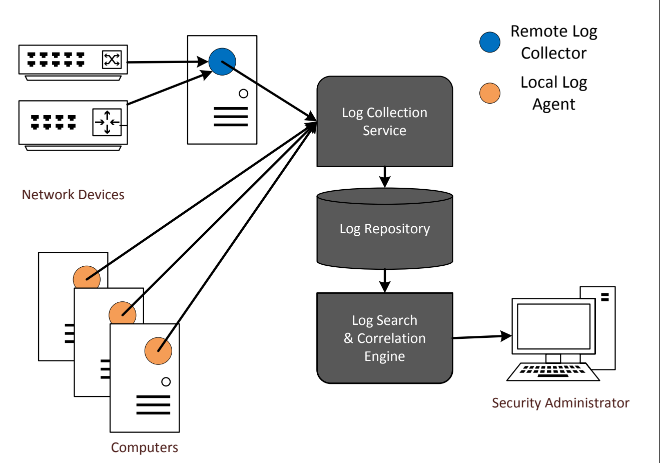

Fig. 1. SIEM Architecture Example

B. Limitation of Current SIEM Technology

Although event detection is a powerful function of SIEM systems for security [1], in many cases, it is largely under-utilized [2]. The most likely cause for this underutilization is the high cost associated with the implementation of the correlation and detection functions with sufficiently high accuracy and specificity for security operations. This high cost is caused by inherent factors such as the complexity and inflexibility of rule-based detection algorithms, and the deterministic parsing schemes in which only logs that follow common logging protocol are properly processed.

Beside the high cost of implementation, the current SIEM technology has a critical limitation of ignoring the information from the unstructured (free-text) part of a log message. We will illustrate this through an example Windows Event Log Entry in Figure 2 below. The unstructured message is the grey-highlighted portion starting from "The IP address..." The structured parts are meta-data added by the Windows Event Log framework, such as "Error," "3/10/2011 2:29:01 PM."

```
Error    3/10/2011 2:29:01 PM      Microsoft-Windows-
Dhcp-Client      1002     Address Configuration State
Event    The IP address lease 10.18.25.108 for the
Network Card with network address 0x801934C9D8E9 has
been denied by the DHCP server 10.5.18.11 (The DHCP
Server sent a DHCPNACK message).
```

Fig. 2. Unstructured Part of a Windows Event Log

It is obvious that grey part contains vital information that is relevant to the error such as IP address of DHCP server involved, the response code, and the rejected IP address, etc. However, this information is ignored in most current SIEM implementations, since the correlation and detection rules utilize only the structured parts which are parsed and stored by a Windows Event parser. The current deterministic log parsing scheme is not capable of handling this type of free-text pseudo natural language. This causes a significant limitation on the types of event detected through SIEM and the depth of information used for correlation of different events.

C. Research Objective and Contribution

This research explores the possibility of exploiting hidden information within the unstructured log messages through text mining and natural language processing. This approach has advantages over the conventional rule-based correlation and detection in several ways. First, by using the information that has been previously ignored we have more relevant information and, therefore, have better chance in increasing accuracy of detection. Second, it is possible to develop a generic parser which can recognize key information in the logs which do not follow common logging protocols, such as application logs. This will further expand the knowledge for event correlation and detection. Third, the detection rule can be generated through supervised or unsupervised learning and can self-adjust to changes in the environment, minimizing the time-consuming process of initial configuration and the subsequent updates.

Nevertheless, it should be emphasized that this approach cannot replace the conventional SIEM altogether. For the critical log management functions, such as log aggregation and archiving, SIEM is the state-of-art technology as of now. Also, many events can be effectively detected through rule-based detection scheme using structured logs with common logging protocols.

The objective of this study is to suggest an avenue for improving the effectiveness and utilization of security information which are collected and aggregated by SIEM, and to open up the possibility of security data analytics using log data. Therefore, the results should not be interpreted in perspective of proving the comparative superiority of this approach to an existing security solution, but of introducing a novel way of complimenting current security technologies.

The contribution of this study lies on the systemic investigation of data preparation, feature extraction and text classification on system-generated log messages. The experiments are conducted on the published SKAION 2006 IARPA Dataset [6, 7]. The tools, settings, and other factors affecting the results were disclosed in detail in order to ensure the repeatability. The resulting performance exhibits a number of important factors affecting data mining of unstructured log messages. These are the types of classification algorithms that work best for a log data format; the effect of training set size to the classification performance; techniques affecting feature extraction and its performance; and the degree of performance gain through feature extraction. It is first attempt, to our knowledge, to apply text classification on published security attack data. It is also first study to use named entity extraction technique to extract security relevant information from system-generated log data.

II. Similar Works

One of the early studies of unstructured log analysis was done by Qiang Fu et al. [3]. They introduced an algorithm to detect the execution anomalies through unstructured logs of Hadoop and SILK. The main difference of this study to ours is that Fu et al. use regular expression to extract specific "log keys" which are predefined patterns based on specific applications. That is, the information extractor already knows what to look for. On the other hand, our approach focuses on extracting "all relevant information" for detection from any unstructured log using natural language processing. This generality is a key concept, not a by-product in our research.

Wei Xu et al. also presented an application of using data mining and statistical learning methods to detect abnormal execution traces from console logs [4]. They used frequent pattern mining and distribution estimation techniques to discover a dominant pattern, and then, used principal component analysis for anomaly detection. An unusual approach of this work is that the authors suggest the analysis of source code to eliminate the uncertainty in parsing application logs. Although this method will guarantee highly accurate results in unstructured log analysis, it cannot be easily adopted to the general cases where the log analyzer does not have access to the source code.

Azodi et al. presented a method to improve IDS/SIEM performance by detecting the input log type and format using regular expression and normalizing log entries [5]. The philosophy behind this approach is very similar to ours. The two main differences are: that Azodi et al. used regular expression whereas we used natural language processing; and that their normalized logs are still fed into rule-based detection engine, while we used machine-learning techniques to detect threats.

III. SKAION 2006 IARPA DATASET

The dataset used for this study is from the Packet Clearing House SKAION 2006 IARPA Dataset [6, 7]. This dataset consists of various logs and network traces captured from a simulated network environment, where benign user activities and malicious attacks are emulated by computer programs [6]. The malicious attacks are of various levels of sophistication ranging from a simple CGI Overflow to attacks involving email phishing [7]. The dataset includes background data which contains the normal level of background activities, including probing and unsuccessful attack attempts. The distribution of these background activities are statistically modeled after the traffic observed at the Air Force Research Laboratory [6].

The total size of the dataset is 119.2 TB, and a large portion of it contains network traffic traces. In our research, approximately 15 GB of text data from the release 4 is used. It consists mainly of logs collected from 136 sources for different attack scenarios and background traffic.

Release 4 includes data from ten different attack scenarios. Since the types of logs collected are not consistent among all ten scenarios, only the five shown in Table 1 were included in this study. Attacks 4s1, 4s3 and 4s4 have a background attack scale of 50%. The background attack scale indicates the percentage of similar attempts recorded in background data. This intentionally introduced noise makes it harder to distinguish the alerts and log entries of these four attacks from that of background data, and demands the experimental results more closely approximate the real world.

TABLE I. ATTACK SCENARIOS

Attack ID	Attack Scenario	Attack Scale	Description
4s1	CGI Overflow	50%	Attacker passes an overflow string to a CGI script on webserver
4s3	CGI Overflow with Decoys	50%	Same as 4s1, but there are many decoys that produce the same footprints in IDS before and after the attack
4s4	Word Macro Exfiltratio n	50%	Attack involves a Word document with a malicious macro sent through email. The macro is activated by one user and uploads all files in the “Recent Files” list to a remote ftp server
4s13	Firewall Misconfig uration	None	An administrator accidentally brings down the firewall, allowing unauthorized traffic to get through to the internal network for a couple of minutes
4s14	Phishing and PNP	None	A user is lured to register a malicious website and he uses the same username/password to the Windows machine on the network. The attacker ssh to the Windows machine and downloads a PNP exploit executable, gaining a command shell. The attacker then uploads all files to a remote ftp site.

IV. TEXT CLASSIFICATION OF UNSTRUCTURED MESSAGES

The problem of detecting malicious activities using unstructured log messages can be seen as a problem of text classification. If we have a classifier that can determine with reasonable accuracy whether a given log message is from normal data or intrusion data, then we can assume that the same classifier can predict the class of an unseen future log entry as well. To build such a classifier, we would need to extract the right information from the logs and select the right classification algorithm to be trained.

There have been many previous studies on the effectiveness of classification algorithms and techniques for text classification of standard natural language corpus [8, 9, 11, 12, 13], scanned OCR documents [14], or social media data [15, 16, 17, 18]. System-generated messages, however, have a few different characteristics to the natural language text that was analyzed extensively in the aforementioned studies. Some of these characteristics are:

- A large portion of the message is repeated many times in a set of log entries
- The number of natural language words used in the text are relatively small
- Actual vocabulary size is large since log messages contain a large number of tokens that are not words, but numbers or codes such as name of executables, status codes, error codes, and numbers in binary, octal, or hexadecimal formats.
- The messages may not follow standard grammar rules

Therefore, a careful examination is required in selecting the right algorithms and features for the classification of machine-generated unstructured messages. With these differences in mind, we approached this study in following three steps:

1) Apply different classification algorithms on message text and measure the performance of each algorithm. This will help determine which algorithm performs well on machine-generated unstructured messages. We can establish this measurement as the baseline.
2) Identify features that are useful for threat detection and extract those features from the unstructured message. Repeat the same classification experiments as in (1) using the extracted features only, and both extracted features and the message together.
3) Analyze the results to determine the classification algorithms and other text classification techniques which improve performance and time for threat detection.

TABLE II. ENTITY OF INTEREST

Entity	Description	Used for Classification
TIME	Date or time: year, month, day, hour, minutes, second, pre/post fix (AM, PM) or time zone	No
APP	Any component of a program or system: o/s, application, session, function name, etc.	Yes
USER	User name, email address or other string containing user name	Yes
HOST	Host name, computer name or IP address	Yes
KEYWORD	Words indicating the state or event: success, error, completed, started, failed, etc.	Yes

V. FEATURE EXTRACTION FROM UNSTRUCTURED MESSAGES

Feature extraction is the process through which we can attain more obscure information from unstructured logs. In this section, we describe three different techniques used to extract relevant information from free-text log messages: Meta-data, Named Entity Recognition (NER), and Log Type Classification.

A. SKAION Log Meta-data

Meta-data for a log message collected by SIEM typically consists of the time of collection, source device, source application, collection agent ID, and other information related to the collection and transfer of log entries to the central repository. Some meta-data, such as the source device and application are important features in correlation and event detection. The SKAION logs are collected off-line using scripts [6] and, thus include little meta-data. During aggregation and sampling of the log entries, some meta-data are kept for reference purposes and two of them, log source and message length, were used for classification.

B. Named Entity Recognition (NER)

Named entity recognition in natural language processing refers to the process of recognizing (or tagging) a sequence of words in a text that are names of things, such as people and company names [23]. In our context, we are interested in recognizing the things that may help us to detect the security threats. For example, user name, application name, host name, IP addresses, or any keywords indicating the status of the application. In order to extract relevant information from unstructured log messages, we first identified the category of things to be recognized, and created the training data by manually tagging a set of sampled logs. Table II describes such entities of interest.

TABLE III. FEATURE EXTRACTION PERFORMANCE - STANFORD NER

Entity Class	TP Rate	FP Rate	Precision	Recall	F-Measure
APP	0.9853	0.01238	0.9789	0.9853	0.9821
HOST	0.9966	0.00073	0.9966	0.9966	0.9966
KEYWORD	0.979	0.00324	0.9906	0.979	0.9848
TIME	1	0	1	1	1
USER	1	0	1	1	1
W. Avg.	0.9886	0.0055	0.9886	0.9886	0.9886

TABLE IV. LOG TYPE CLASSIFICATION PERFORMANCE – SMV

Class	TP Rate	FP Rate	Precision	Recall	F-Measure
AUDIT	0.998	0.013	0.986	0.998	0.992
INFO	0.971	0.007	0.966	0.971	0.968
ERROR	0.833	0.003	0.938	0.833	0.882
WARN	1	0	1	1	1
ALERT	0.984	0	1	0.984	0.992
W. Avg.	0.985	0.007	0.985	0.985	0.985

The Conditional Random Field (CRF) based classifier from the Stanford NLP library [23] is used to create and train the model. Tested on ten separately sampled test data, the classifier tagged the interested entities with an average accuracy of 0.9886. Table III shows the average performance per entity category. Using a CRF-based statistical NER system to extract entities has advantages over using simple character pattern matching through regular expression. The CRF Classifier models the sequence of words, rather than individual words separately. Therefore, it recognizes the patterns of words occurring around the entity we are interested in. This allows the classifier to also recognize the entity by the patterns of the sequence containing it, even if the word itself does not exactly match the pattern of the character string seen in the training data.

C. Log Type Classification

Log type is a generic classification of log entry that indicates the purpose or priority level, such as information, error, audit, warning, etc. Log types are useful in threat detection. For example, we can construct and test a hypothesis such as: "if the distribution of certain log type changes in a set of log entries, the target system is not in a normal state".

Two problems exist in the SKAION dataset to determine log types reliably. First, some free text logs do not have any log type information. Second, even if there are certain log types provided by the logging protocol, there is no reliable way of normalizing them into a set of standard categories. For example, Windows Event Type 1180 cannot be translated to a category that is corresponding to a Syslog Priority level, ERROR. Therefore, we cannot reliably interpret the real importance, or priority based on these meta-data.

To overcome these problems, we ignored the meta-data and determined the log types from the free-text messages using Support Vector Machine (SVM) classifier. 200 log entries were sampled from each of class and manually labeled for training. In a standard 10-fold test using WEKA machine learning library [22], an average of 98.5% log messages were labeled correctly. (See Table IV)

VI. EXPERIMENT SETTINGS

A. Tools, Libraries, and Programs

A custom-developed application written in Java is used to perform all experiments described in this section. Other than JavaSE-1.8, the following external libraries and tools were used in the various stages of the experiment and analysis: Stanford NER 3.5.2; WEKA v.3.6.13; R v. 3.2.0; Rattle v.3.4.1

B. Log Aggregation and Sampling

The collection of log data was pre-processed and aggregated into six different consolidated log data files: one per attack scenario, and one background data. Each line of the data file contains one log entry, including a text message and meta-data such as the source file name, line number and message length. Each entry is also labeled with an attack ID. The aggregated files are then parsed for feature extraction using NER and the SVM Log Type Classifier.

To train and test the classifier, random samples of size ranging from 500 to 2000 were selected from each attack type, and samples of the same size were selected from the background data. For example, a data file of sample size 500 contains 500 log samples from one of the attack types and 500 log samples from the background traffic. The sampled logs are further processed into three different datasets:

1) *Message-only data:* contains labeled free-text log messages without any meta-data. Within the message, any string indicating time or date is deleted, in order to remove the strong correlation between the label and the time variable. This correlation is intuitive since the log collection for a particular attack is a snapshot at a particular time during that attack.
2) *Features-only data*: contains the extracted features (user, application name, host, keywords) and meta-data (message length, source system) without any original message. All features indicating time or file locations are removed.
3) *Message-and-Features data:* contains the extracted features, meta-data, and free-text messages. Both time/date strings within the message and the time variable extracted from the messages are removed.

C. Performance Evaluation

Following five statistics were used to evaluate the classification performance throughout the experiments:

- Accuracy = $\frac{\sum True\ positive + \sum True\ negative}{\sum Total\ Population}$
- Precision = $\frac{\sum True\ positive}{\sum Classified\ as\ positive}$
- Recall = $\frac{\sum True\ positive}{\sum Positive}$
- Specificity = $\frac{\sum True\ Negative}{\sum Negative}$
- Time = elapsed time for training and 10-fold testing

Recall and Specificity are negatively related. Therefore, there would be a trade-off if we focused more on one of these two measures. For our evaluation, we are biased for the small variance among the four performance metrics – accuracy, precision, recall and specificity. A larger variance means that a good performance in one of these measures may lead to a larger error in the other measure. We also preferred the shorter elapsed time for obvious reasons.

The performance metric reported is an average value from 10-fold testing with 66% -33% split. Time measures the elapsed time for training and 10-fold testing in milliseconds.

VII. Performance Analysis

A. Text Transformation

In order to classify a log message using a classification algorithm, it has to be transformed into a numeric vector. Without any specific feature extraction, the message can be transformed into a word vector representation, [8] where each log entry is represented as a vector of bits or integers; the number indicates whether the entry contains a specific word (a bit) or the frequency of occurrence of the word in the message (an integer). In our experiments with the SKAION dataset using the WEKA machine learning tool [22], some common text transformation techniques such as stemming, removing stop words, or using n-gram features showed no positive impact on the performance of the classifier. On the other hand, TF-IDF transformation [8] has a small positive impact (.23%) on classifier performance, as long as it is used with appropriate attribute selection strategies. (Figure 3).

B. Attribute Selection

Attribute Selection is a technique for reducing dimensionality by removing non-informative attributes selectively. Yang and Pederson report in their comparative study, attribute selections based on Information Gain (IG) and the chi-squared test (Chi) are most effective for text classification [9]. In our experiment with the SKAION dataset, the Information Gain and chi-squared test attribute selection schemes increased the performance and reduced the training and testing time by a similar level.

Figure 3 and 4 show the relative gain of time and accuracy for a chi-test attribute selection. The performance metrics using attribute selection (tf-idf_and_chi and tf-id_and_ig) in Figure 3 shows 2.7% higher precision and 30% higher specificity, in comparison to the metrics for not using attribute selection (no-tf-idf_no_select). Attribute selection also reduced the time for

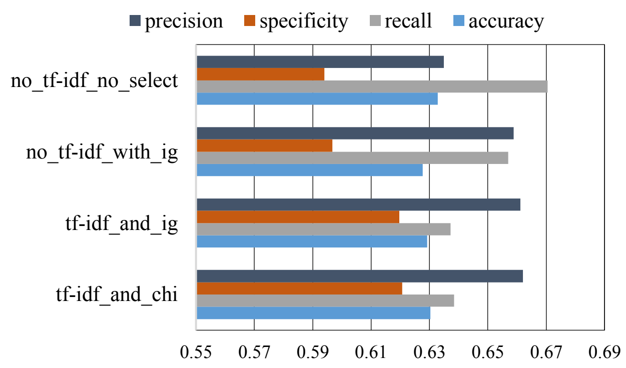

Fig. 3. Average Performance TF-IDF and Attribute Selection

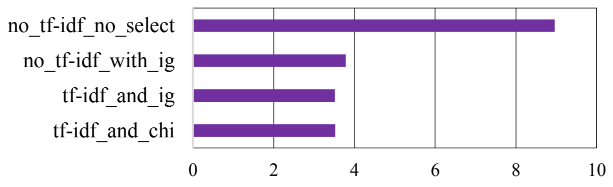

Fig. 4. Average Time in seconds for Training and 10-fold Testing

training and testing by 63% as shown in Figure 4, by removing approximately 90% of variables with low prediction value.

C. Performance By Classification Algorithms and Data Format

In order to find the best performing classification algorithms for unstructured log analysis, a number of experiments were conducted with 12 different classifiers from the WEKA library [22] over 3 different formats of datasets: message-only, features-only, and message-and-features. It should be noted that all experiments used the same sample log entries, although the input data were prepared in different formats.

The results indicate that different classification algorithms were effective on different input formats. Table V shows that the Naïve Bayes Multinomial (NBM), Voted Perceptron (VP), and AD Tree (ADT) algorithms (shown in red) perform best for message-only data. These three algorithms display high values of performance statistics with little variances among them. For features-only data, (Table VI) Bayes Net (BN), Random Forest (RF) and Random Tree (RT) show strong performance. For data with both messages and extracted features (Table VII), tree-based algorithms such as Random Tree (RT), J48 Tree (J48), and Random Forest (RF), perform better than others.

Overall, the best performance is achieved when input data include both message and extracted features. As shown in Table VII, the maximum average performance reached 0.7673 for message-and-features dataset, compared to 0.6766 and 0. 6539 for feature-only and message-only datasets, respectably. The performance gain comes with the cost of time in this case, since it takes an average of 19.6% longer than classifying message-only data and 140% longer than features-only data.

A significant time gain is observed on features-only dataset for all classification algorithms. This is due to the reduction in the number of attributes. Free-text messages yield about 120-160 attributes after the attribute selection in 1000 sample log entries; whereas features-only data has only 12-18 attributes for the same size of input. This resulted in 78.50% time reduction in average in comparison to message-only dataset. Moreover, the extracted features contain less noise and, therefore show an average 3.48% performance gain.

D. Performance by Sample Size

In order to find the relationship between the size of training data and the classification performance, we ran the classification experiments with four different sample sizes: 500, 1000, 1500, and 2000. The samples were formatted to include both extracted features and free-text messages. Random Tree classification algorithm was used because of its relatively stable performance and speedy execution for all data formats.

Figure 5 displays the average performance for all five attack types. All performance statistics increase from 500 to 1500, but fall at 2000. The reason for the degradation at 2000 is not clear. The cause could be that the classifiers reached their maximum predictive power on a given dataset between 1500 and 2000, or that the WEKA software reached its maximum capacity in terms of input data size. This issue is discussed further in Section IX. Time, shown in the purple line, seems to increase exponentially.

TABLE V. PERFORMANCE BY CLASSIFICATION ALGORITHMS – MESSAGE-ONLY

	Acc.	Recall	Sepc.	Prec.	Mean	StdDev	Time
SMO	0.6373	0.5819	0.6910	0.7054	0.6539	0.0562	1862
KStar	0.6293	0.6828	0.5748	0.6391	0.6315	0.0444	24293
JRip	0.5965	0.6296	0.5611	0.6040	0.5978	0.0282	3705
PART	0.6158	0.6458	0.5835	0.6428	0.6220	0.0290	2891
J48	0.6187	0.6742	0.5609	0.6344	0.6221	0.0470	1431
NMB	0.6313	0.6168	0.6440	0.6652	0.6393	0.0205	544
NB	0.5904	0.5492	0.6352	0.6267	0.6004	0.0392	814
BN	0.5916	0.4571	0.7240	0.7360	0.6272	0.1309	725
VP	0.6280	0.6069	0.6481	0.6507	0.6334	0.0204	491
ADT	0.6253	0.6399	0.6077	0.7078	0.6452	0.0438	5567
RT	0.6191	0.7131	0.5241	0.6129	0.6173	0.0772	719
RF	0.6230	0.6735	0.5711	0.6240	0.6229	0.0418	27013
MAX	0.6373	0.7131	0.7240	0.7360	0.6539		
AVG							5838

TABLE VI. PERFORMANCE BY CLASSIFICATION ALGORITHMS – FEATURES-ONLY

	Acc.	Recall	Sepc.	Prec.	Mean	StdDev	Time
SMO	0.6380	0.6735	0.6008	0.6441	0.6391	0.0299	2822
KStar	0.6396	0.6786	0.6003	0.6305	0.6373	0.0323	3612
JRip	0.6312	0.6321	0.6286	0.6418	0.6334	0.0058	543
PART	0.6388	0.5969	0.6802	0.6558	0.6429	0.0351	583
J48	0.6430	0.6616	0.6236	0.6416	0.6425	0.0155	126
BN	0.6406	0.6455	0.6331	0.6748	0.6485	0.0183	107
NB	0.6394	0.6310	0.6456	0.6853	0.6503	0.0241	126
ADT	0.6333	0.5186	0.7466	0.6859	0.6461	0.0968	629
RF	0.6748	0.6563	0.6932	0.6821	0.6766	0.0155	3870
RT	0.6620	0.6632	0.6603	0.6629	0.6621	0.0013	127
MAX	0.6748	0.6786	0.7466	0.6859	0.6766		
AVG							1255

TABLE VII. PERFORMANCE BY CLASSIFICATION ALGORITHMS – MESSAGE-AND-FEATURES

	Acc.	Recall	Spec.	Prec.	Mean	StdDev	Time
SMO	0.6695	0.7105	0.6270	0.6692	0.6690	0.0387	4263
KStar	0.6539	0.7028	0.6047	0.6441	0.6514	0.0181	28340
JRip	0.6479	0.6095	0.6847	0.6726	0.6537	0.0341	4290
PART	0.6743	0.6431	0.7050	0.6882	0.6776	0.0403	5553
J48	0.6810	0.6721	0.6892	0.6888	0.6828	0.0332	1086
NB	0.6107	0.6047	0.6172	0.6456	0.6196	0.0262	694
BN	0.6348	0.6148	0.6523	0.7050	0.6517	0.1616	782
ADT	0.6683	0.4924	0.8433	0.8177	0.7054	0.0081	6819
RT	0.6794	0.6761	0.6826	0.6807	0.6797	0.0148	679
RF	0.6976	0.6800	0.7151	0.7050	0.6994	0.0027	18604
MAX	0.6976	0.7105	0.8433	0.8177	0.7673		
AVG							7111

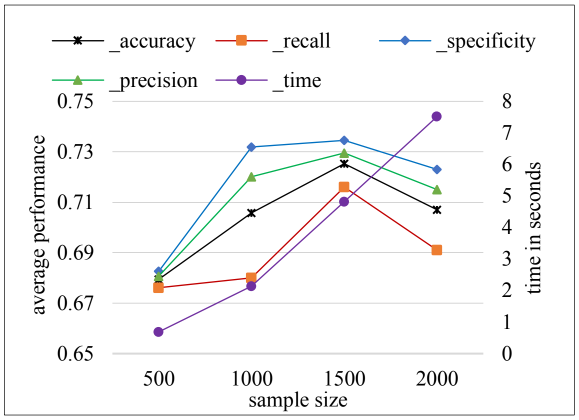

Fig. 5. Performance By Sample Size

E. Performance Gain by Feature Extraction

In this section, we examine more closely the performance gain attained through feature extraction in comparison to the baseline performance of direct text classification of fee-text message.

As discussed in Section V, feature extraction requires additional processing of running raw log entries through NER and SVM Log Type Classifier. During the real-time processing of log entries in a SIEM system, this process will add extra time in parsing incoming logs and saving the extracted information. It is difficult to say exactly how much this process slows down SIEM processing because the speed varies depending on the hardware configuration and the baseline performance. In our experiments with SKAION dataset, for example, the feature extraction module processed 9995.36 words per second in average on a 2.50 GHz processor with 16 GB memory. Nevertheless, this extra time is not negligible when we consider the velocity and volume of typical log collection; and therefore, cannot be justified without the performance gain.

In order to measure the performance gain by feature extraction more precisely, we ran an experiment with the techniques and settings that showed the best average performance by each data formats. That is, NBM and VT classifiers for message-only dataset, RT and RF for feature-only dataset, and J48 and RT for the dataset with message-and-features. We used a sample size of 1500 which had given the best measures in previous experiments. For all data types, we used TF-IDF measures and IG attribute selection as it is shown to improve performance.

The result shows that the features-only dataset, containing only 19 to 33 extracted attributes, ran 87% faster than message-only dataset (Figure 8), and achieved 16.73% accumulative performance gain (Figure 6, message to features) in comparison to the message-only datasets. Still, the dataset with message-and-features recorded the higher rate of accuracy, recall, specificity and precision. Compared with message-only data, it gained an average of 6.82% accuracy, 8.49% recall, 2.86% precision, and 5.30% specificity. This is a 23.5% cumulative gain (Figure 6, message to both). However, this gain in performance came with the cost of time. As shown in Figure 13, the data set with both features and messages took approximately 95% longer to build a model and perform a 10-fold test.

The results imply that it is more beneficial to perform feature extraction than to classify free-text message directly since 87% gain in classification time and 16.73% gain in accuracy are most likely to outweigh the extra time for feature extraction process. Moreover, the fact that message-and-features datasets result in better performance metrics indicate that there is still more information in the free-text message to be extracted.

The Receiver Operating Characteristic (ROC) curves in Figure 7 are consistent with the previous analysis. The curve for the dataset with messages-and-features (red) exhibits the largest area under the curve, signifying the strongest predictive power. The features-only data type (blue) is next, while the message-only data type (purple) has the smallest ROC area.

VIII. Summary of Results

From various experiments in this study, we know that the following factors affect the classification performance for unstructured log analysis:

1) Identifying, extracting, and using features such as the application name, host name, IP addresses, user name, and keywords from the unstructured log messages increase the classification performance by 2.8-8.5% and decrease time by 87%.
2) The classification algorithm's performance is significantly affected by the nature of the data preparation. If the prepared data contains only free-text log messages, algorithms such as NBM or VT work well. If the data contains categorical features or is a mixture of features and free-text messages, RT or RF show a robust performance.
3) The attribute selection algorithm based on Information Gain or the chi-squared test increases the average precision by

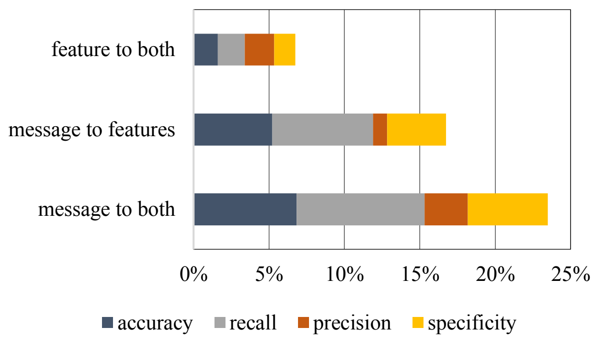

Fig. 6. Performance Gain By Feature Extraction

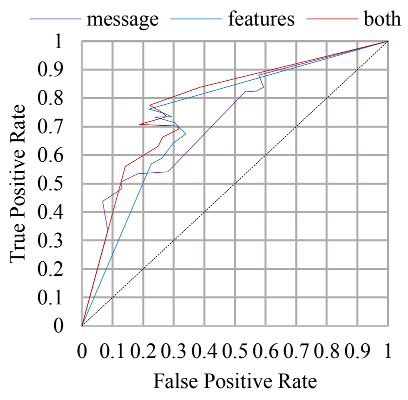

Fig. 7. ROC curve

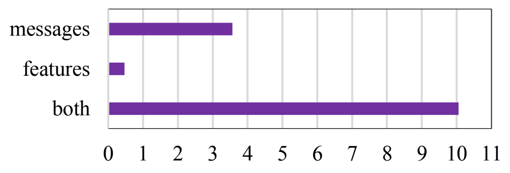

Fig. 8. Average Time in seconds by Data Type

2.5% and decreased the average recall by 3.0%. This also reduced time for training and testing by 63%.

4) Generally, the size of the training data is positively related to the classification performance, until the peak is reached. This peak was observed in the SKAION dataset between the sample sizes of 1500 and 2000 for binary classification.
5) TF-IDF transformation of the free-text messages has a small but positive impact if it is used with the proper attribute selection algorithm.

IX. Discussion

Through this study, we discovered many factors that affect unstructured log analysis using machine learning. Still, the results raise some questions. We will discuss some of them in this section.

1) Is the predictive performance of 70-73% enough for security threat detection?

It is difficult to find a reliable SIEM benchmark for security threat detection, because SIEM is primarily used as a log collection and archiving tool as stated earlier. The most common benchmarking metric for SIEM is Event per Second (EPS), which indicates how many log entries a system can handle per second [1]. Since the availability of the KDD99 dataset in 1999, more benchmarking studies have been done on network-based security detection through the Intrusion Detection/Prevention System (IDS/IPS) [24, 25]. Lippmann et al. found 18% of attacks were completely missed by a signature-based Network-based Intrusion Detection System (NIDS) [24]. A more recent study with the same dataset reports detection rates ranging from 5.4 to 99.4%, suggesting that a network-based IDS performs well for certain security threats, but had little success on others [26]. This is because NIDS relies solely on network traffic analysis and has little insight into the events occurring locally within a computer or non-network based attacks such as email phishing or misconfiguration. On the other hand, a SIEM handles information from almost any system. Considering we achieved a 70-73% prediction only using unchartered information within free-text data from diverse range of systems, it is reasonable to conclude that there is high potential for this approach. With further studies on intelligent feature extraction, correlation of other security logs - IDS/IPS alerts, email filter logs, or CMS (configuration management system) logs, the unstructured log mining will be a useful tool to improve the overall security detection rate.

2) Is the performance gain of 2.8-8.5% worth the time and effort of going through feature extraction?

Some statistical analytics experts in business marketing suggest that using non-traditional data sources such as social networks improved their predictions by 0.5-1.5% [27]. When analytics have reached their maximum predictive potential with the existing data and techniques, adding more data to gain another 1 or 2% can add further competitive advantage. Therefore, the 23.5% of cumulative performance gain is not negligible. For our purposes, however, there are two more important benefits of feature extraction, beyond the strict gain in performance metrics:

1) Using features alone, we achieved better prediction performance in only fraction (13%) of elapsed time (Figure 8). This suggests the possibility of real-time unstructured log analysis using classifiers with a continuous learning capability.
2) The extracted features are important keys for the normalization of log messages and the analysis of the relationships between log entries. Normalization of time feature, for example, would enable us to construct a sequence of events from different devices by the generation time, rather than the collection time, providing more accurate timing of events. Another example could be the correlation of one user name appearing free-text log messages from different devices which would allow to trace the user's behavior pattern with greater accuracy.

3) Does the performance peak with the training data size indicate the natural limit for the dataset?

In Section VII-D, we observed that a peak was reached at 1500. If this is the natural limit for this dataset, adding more training data would not improve performance. However, if this limitation is imposed by the software or hardware, then by using different tools, we may further improve the classification performance. The relationship between training data size and performance appears to be highly dependent on the types of datasets being studied [28, 29]. Therefore, further work is necessary to answer this question satisfactorily.

X. Conclusion

This study systematically investigates the techniques for text classification, natural language processing, and machine learning in mining unstructured log messages for the purposes of security threat detection. A number of experiments were conducted on simulated attack data from SKAION datasets. In order to extract the relevant information from the unstructured log messages, techniques for named entity recognition and generic text classification were used. The extracted information was preprocessed into three different formats: free-text messages, extracted features, and both messages and features. Through a number of experiments, the best classification performance metrics (70-73%) were achieved on data including both free-text messages and extracted features by using the Random Tree and J48 Tree algorithms with TF-IDF transformation and IG attribute selection. Using extracted features only, we also achieved a similar 68-71% performance metric in only 5.3% of the time duration. Therefore, feature extraction and text classification of unstructured log messages demonstrate high potential for real-time log analysis using machine learning.

Acknowledgment

This work is sponsored by UNLV Doctorial Graduate Research Assistantship Program. Support for the Packet Clearing House SKAION 2006 IARPA Dataset is provided by the U.S. Department of Homeland Security, Science and Technology Directorate, PREDICT project.

References

[1] ROSA, L., ALVES, P., CRUZ, T., SIMÕES, P. AND MONTEIRO, E. 2015. A comparative study of correlation engines for security event management. In Iccws 2015-The Proceedings of the 10th International Conference on Cyber Warfare and Security, Academic Conferences Limited, 277.

[2] YEN, T., OPREA, A., ONARLIOGLU, K., LEETHAM, T., ROBERTSON, W., JUELS, A. AND KIRDA, E. 2013. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In Proceedings of the 29th Annual Computer Security Applications Conference, ACM, 199-208.

[3] FU, Q., LOU, J., WANG, Y. AND LI, J. 2009. Execution anomaly detection in distributed systems through unstructured log analysis. In 2009 ninth IEEE international conference on data mining, IEEE, 149-158.

[4] XU, W., HUANG, L., FOX, A., PATTERSON, D. AND JORDAN, M. 2009. Online system problem detection by mining patterns of console logs. In Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on, IEEE, 588-597.

[5] AZODI, A., JAEGER, D., CHENG, F. AND MEINEL, C. 2013. A new approach to building a multi-tier direct access knowledgebase for ids/siem systems. In Dependable, Autonomic and Secure Computing (DASC), 2013 IEEE 11th International Conference on, IEEE, 118-123.

[6] Packet Clearing House, SKAION 2006 IARPA Dataset. http://pch.net.

[7] https://www.predict.org.

[8] AGGARWAL, C.C. AND ZHAI, C. 2012. Mining text data. Springer Science & Business Media.

[9] BASU, T. AND MURTHY, C. 2012. Effective text classification by a supervised feature selection approach. In Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on, IEEE, 918-925.

[10] MENG, W., LANFEN, L., JING, W., PENGHUA, Y., JIAOLONG, L. AND FEI, X. 2013. Improving short text classification using public search engines. In Integrated Uncertainty in Knowledge Modelling and Decision Making, Springer, 157-166.

[11] WEISS, S.M., INDURKHYA, N., ZHANG, T. AND DAMERAU, F. 2010. Text mining: predictive methods for analyzing unstructured information. Springer Science & Business Media.

[12] YANG, Y. AND PEDERSEN, J.O. 1997. A comparative study on feature selection in text categorization. In ICML, 412-420.

[13] ZHANG, W., YOSHIDA, T. AND TANG, X. 2011. A comparative study of TF* IDF, LSI and multi-words for text classification. Expert Systems with Applications 38, 2758-2765.

[14] TAGHVA, K. 2009. Identification of Sensitive Unclassified Information. In Computational Methods for Counterterrorism, Springer, 89-108.

[15] GATTANI, A., LAMBA, D.S., GARERA, N., TIWARI, M., CHAI, X., DAS, S., SUBRAMANIAM, S., RAJARAMAN, A., HARINARAYAN, V. AND DOAN, A. 2013. Entity extraction, linking, classification, and tagging for social media: a wikipedia-based approach. Proceedings of the VLDB Endowment 6, 1126-1137.

[16] AGICHTEIN, E., CASTILLO, C., DONATO, D., GIONIS, A. AND MISHNE, G. 2008. Finding high-quality content in social media. In Proceedings of the 2008 International Conference on Web Search and Data Mining, ACM, 183-194.

[17] BECKER, H., NAAMAN, M. AND GRAVANO, L. 2010. Learning similarity metrics for event identification in social media. In Proceedings of the third ACM international conference on Web search and data mining, ACM, 291-300.

[18] PANG, B. AND LEE, L. 2008. Opinion mining and sentiment analysis. Foundations and trends in information retrieval 2, 1-135.

[19] AGGARWAL, C.C. AND ZHAI, C. 2012. A survey of text classification algorithms. In Mining text data, Springer, 163-222.

[20] KOTSIANTIS, S.B., ZAHARAKIS, I. AND PINTELAS, P. 2007. Supervised machine learning: A review of classification techniques.

[21] JOACHIMS, T. 1997. Probabilistic of the Rocchio algorithm with TFIDF for text categorization. In Proceedings of the 14th International Conference on Machine Learning, Nashville, TN, USA

[22] HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P. AND WITTEN, I.H. 2009. The WEKA data mining software: an update. ACM SIGKDD explorations newsletter 11, 10-18.

[23] http://nlp.stanford.edu/software/CRF-NER.html

[24] LIPPMANN, R., HAINES, J.W., FRIED, D.J., KORBA, J. AND DAS, K. 2000. The 1999 DARPA off-line intrusion detection evaluation. Computer networks 34, 579-595.

[25] ELKAN, C. 2000. Results of the KDD'99 classifier learning. ACM SIGKDD Explorations Newsletter 1, 63-64.

[26] HOQUE, M.S., MUKIT, M., BIKAS, M. AND NASER, A. 2012. An implementation of intrusion detection system using genetic algorithm. arXiv preprint arXiv:1204.1336.

[27] HILL, S., PROVOST, F. AND VOLINSKY, C. 2006. Network-based marketing: Identifying likely adopters via consumer networks. Statistical Science 256-276.

[28] BANKO, M. AND BRILL, E. 2001. Mitigating the paucity-of-data problem: Exploring the effect of training corpus size on classifier performance for natural language processing. In Proceedings of the first international conference on Human language technology research, Association for Computational Linguistics.

[29] ZHU, X., VONDRICK, C., RAMANAN, D. AND FOWLKES, C. 2012. Do We Need More Training Data or Better Models for Object Detection? In BMV