5-1-2016

Mining Unstructured Log Messages for Security Threat Detection

Candace Suh-Lee
University of Nevada, Las Vegas, suhlee@unlv.nevada.edu

MINING UNSTRUCTURED LOG MESSAGES FOR SECURITY THREAT DETECTION

By

Candace Suh-Lee

Bachelor of Science, Computer Science
University of Toronto, Canada
2002

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2016

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

April 21, 2016

This thesis prepared by

Candace Suh-Lee

entitled

Mining Unstructured Log Messages for Security Threat Detection

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Yoohwan Kim, Ph.D.
Examination Committee Chair

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Interim Dean

Kazem Taghva, Ph.D.
Examination Committee Member

Justin Zhan, Ph.D.
Examination Committee Member

Yahia Baghzouz, Ph.D.
Graduate College Faculty Representative

Abstract

As computers become larger, more powerful, and more connected, many challenges arise in implementing and maintaining a secure computing environment. Some of the challenges come from the exponential increase of unstructured messages generated by the computer systems and applications. Although these data contain a wealth of information that is useful for advanced threat detection and prediction for future anomalies, the sheer volume, variety, and complexity of data make it difficult for even well-trained analysts to extract the right information. While conventional SIEM (Security Information and Event Management) tools provide some capability to collect, correlate, and detect certain events from structured messages, their rule-based correlation and detection algorithms fall short in utilizing information in unstructured messages. This study explores the possibility of utilizing techniques for text mining, natural language processing, and machine learning to detect security threat 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 the performance of prediction is evaluated. In comparison to the base case without feature extraction, an average of 16.73% of accumulated performance gain and 84% of time reduction was achieved using extracted features only, while a 23.48% performance gain with 82.39% of time increase was attained using both unstructured free-text messages and extracted features. The results display strong potential for further increase in performance by using larger size of training sets and extracting more features from the unstructured log messages.

Acknowledgments

I would first like to express my gratitude to my advisor, Dr. Yoohwan Kim for his excellent guidance and assistance throughout my study. I would also like to thank Dr. Kazem Taghva, Dr. Justin Zhan, and Dr. Yahia Baghzouz for their support. It is an honor to have them in my thesis committee.

I am very grateful to Cassandra Lee for her enormous help in editing this thesis. Lastly, my deepest gratitude goes out to my husband, Eric Eunmok Lee for his constant support and encouragement. This work would not have been possible without his countless sacrifices.

Candace Suh-Lee

Table of Contents

List of Tables

List of Figures

List of Listings

Chapter 1. Introduction

1.1 Security Information and Event Management (SIEM)

The term Security Information and Event Management (SIEM) is fairly generic, and seemingly referencing any technology or practice that aims to manage any information and events related to information security. However, in our current security technology landscape, this term commonly refers to one specific type of technology whose main functions are 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.

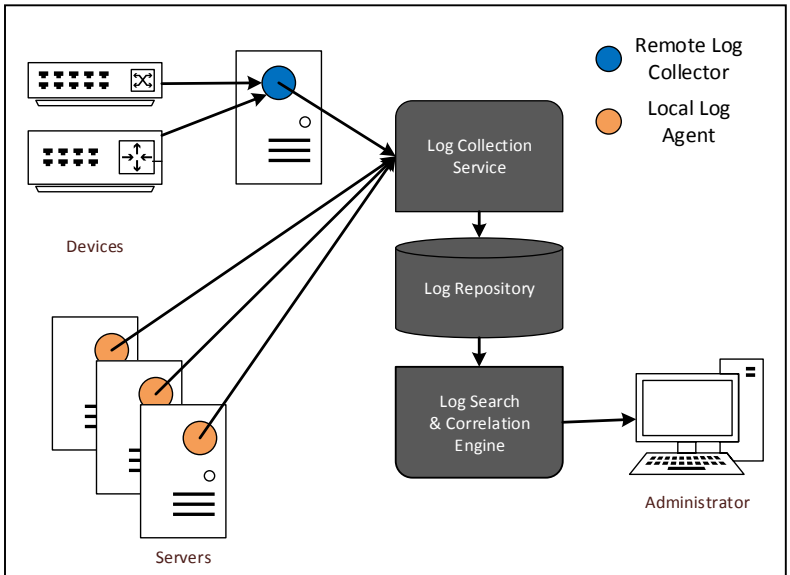
The functions of SIEM become essential in an architecture where applications are distributed among many specialized devices. In the architecture of the earlier computing era, where there was one central computer and many consoles connected to it, almost all important messages were logged centrally and the administrator could easily find all the sequences of transactions in one log repository in that central computer. However, this is not the case in most computing environments today. For example, to support one small commercial web application, we usually provision at minimum two webservers, one to two application server(s), and two database servers for redundancy and load balancing. Besides servers, there would be a few desktops or laptops and wireless devices connected to the system, along with several switches, firewalls, and routers to connect them all. If all these devices generated logs that are meant to be used for support, maintenance, security, or audit, and stored them locally, we can easily see the inefficiencies involved with accessing each device individually and digging for the right information. We can

also imagine what it might be like to run a few hundreds or thousands of devices connected in a large computing environment.

The amount of messages generated is also an important justification for SIEM. The devices in our connected environment today tend to be small but very chatty, because many applications running on these devices need to log additional inter-device communications. This log verbosity could be adjusted for some logs, but not all. For example, many financial transaction logs are required to be generated at a level that can provide complete audit traces of an auditable transaction and must be stored for many years for compliance reasons. With limited memory and storage in these devices, it makes sense to store only small portions of recent logs locally and send the majority to a central location for storage and archiving.

The most popular justification for SIEM has to do with the concept of correlating logs that are collected from many dispersed devices in order to detect events that are normally unnoticeable if each log is checked separately. This is the reason SIEM has "Security" in its name, rather than being called a simple log management system. This function is 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.

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



[Listing 1] Windows Event Log Entry - Example

[Figure 1] SIEM Architecture - Example

1.2 Limitations of Current SIEM Technology

Although event detection is the most unique and powerful function of SIEM systems for security [27], in many cases, it is largely under-utilized [39]. The most obvious cause for this lack of implementation is the high cost associated with utilizing correlation and detection features with sufficiently high accuracy and specificity for security operations. This high cost is caused by indirect causes such as complexity and inflexibility of rule-based detection strategies, and deterministic parsing schemes in which only certain logs that follow a specific logging protocol are understood.

The complexity of detection rules stems from the rule-based correlation/detection strategy employed by most traditional SIEM systems. The correlation/detection rules are pre-configured based on knowledge of previous attacks and their log traces. We will illustrate this through the following example: One very common rule accompanying most SIEM systems is “to create an

alert if there is a successful login after some number of consecutive failed login attempts." This rule is intended to detect a password-guessing or brute-force attack and when active, triggers an alarm for an operator to investigate further. The limit for failed login attempts, the time window for "consecutiveness," and some other parameters need to be configured by the administrator. This sounds simple enough if we have a few machines in one location. But if we have a few thousand machines across the globe that are used by people with a diverse range of technical abilities, this problem becomes much more complex. First, we must decide which devices should have this rule active and determine the number of average failed login attempts for normal usage on each of these devices. Due to the fact that most organizations do not have this type of information on hand, the system must first be run for a period of time using default values. During this tuning period, the operator is required t to investigate each alert generated by this rule and label it as normal or malicious. When this tuning period ends, all alerts generated by the rule must be analyzed, thresholds must be adjusted, and the tuning process needs to be repeated. This tuning process is not completed until nearly all alerts triggered by this rule are indeed malicious. If all this effort is required for one rule, then we can imagine the effort required for a few hundred rules and the thousands of alerts generated by them.

Another limitation of rule-based detection is the fact that it is not adaptable. As new threats and attack tactics are discovered, SIEM rules also need to be updated to detect these new threats. Unlike anti-viruses where the new signatures can be injected remotely to all instances, SIEM's rules are heavily dependent on the environment's architecture and the applications existing within it. For the rule "successful login after a number of unsuccessful attempts" in the previous example to work, the device needed to run the Windows O/S and a member of a Windows domain. If we want to correlate the rule with logs from Ubuntu Linux servers, then a number of custom rules

must be written. If we added a few more Solaris Linux servers into the environment, the custom rules may need to be further re-written, tested and tuned. The larger the organization, the more changes occur daily - new users are added, devices are removed, traffic is rerouted, and vendors come and go. These changes require almost constant reconfiguration and adjustment of the SIEM rules in order to maintain effective operations.

The second reason for the under-utilization of the log correlation functionality of SIEM is deterministic parsing schemes. Parsing is the process by which SIEM systems read log entries and populate databases, so that the database can then be queried by the rules. Since the parser needs to understand different parts of log entries, a specific parser is used for a specific logging protocol. Popular logging protocols such as SNMP, Syslog, or Windows Event Logs are processed through SNMP parsers, Syslog parsers, etc. The contents of less common or less structured logs such as many application logs are largely ignored and stored as free-text content with some meta-data only. In order to parse an unstructured or uncommon log properly, a custom parser must be developed. In order to use the data stored by the custom parser, new database structures, new queries, and/or new correlation rules also need to be developed. The complexity and inflexibility of these processes drive up the cost for fully utilizing the SIEM's log correlation function.

1.3 Unstructured Messages and Hidden Information

Unstructured messages are free-text contents in log entries that are generated by software. These are actual messages to the reader regarding the status of the program which were written by the programmers and included as useful information for the users and administrators of the system.

For example, in a Windows Event Log Entry message in Listing 1 below, the unstructured message is the grey-highlighted portion starting from "The IP address..." It is not difficult to see

```
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).
```

[Listing 2] Windows Event Log Entry - Example

that this section contains vital information that is relevant to the error. The structured parts are meta-data added by the Windows Event Log framework, such as "Error," "3/10/2011 2:29:01 PM" above. In most current SIEM implementations, the correlation/detection rules only utilize the structured parts of the log entry such as the time, source, and event type, and ignore most information stored in the unstructured message. This is due to the SIEM vendor's preference to develop rules that work out-of-box in almost all instances. If a rule uses only meta-data, it is almost guaranteed to work with any logs using Windows Event Log facilities, which avoids the high cost associated with customization.

This approach, however, puts a significant limitation on what we can do with the information stored in the log for obvious reasons. In the given example, all we can work with is the information that at 1:29:01 PM, the Microsoft-Windows-Dhcp-Client has an error with Event ID 1002-Address Configuration State Event. We cannot use the information in the unstructured part, such as the IP addresses, MAC addresses, or the error code from the DHCP Server.

The difficulty of parsing unstructured messages comes from the fact that they are at least partially written in natural language. These phrases are formed by the human developers of the program in order to communicate with the human users of the system. Therefore, the challenges

regarding the automatic processing of logging messages also partially involve natural language processing.

1.4 Research Objectives and Contributions

This research explores the possibility of harnessing recent developments in machine learning in order to exploit the hidden information within unstructured messages to detect events. This is to augment the limitations of current SIEM technology and the experiment results would contribute the improvement of limitations mentioned above. For example, if the detection rules are automatically generated through supervised or unsupervised learning and can be self-adjusted to changes in the environment, then the time-consuming process of initial configuration and subsequent updates can be minimized. Also, if there is a generic parser that can recognize and understand key information in uncommon or unstructured logs through the techniques of text mining and natural language processing, the task of custom parser development for such logs could be reduced to the simple task of training the parser with the sample logs of the environment. These two improvements, if properly implemented, would eliminate a large portion of manual coding and tuning, resulting in increased accuracy and a reduced cost.

Moreover, we believe this approach has the benefit of pushing the limits of the traditional SIEM by utilizing information that is currently ignored. We hope that the extra information will not only improve the performance of SIEM, but also be able to detect events that were previously impossible to notice due to a lack of information. There is also the potential to be able to detect anomalies which were previously unseen by means of pattern recognition and auto-tuning.

Nevertheless, it should be emphasized that this study alone in no way resolves all the problems and improvements discussed above, nor does it provide an alternative solution to the

current SIEM. This study is a preliminary step towards using machine learning in order to take advantage of information that is already collected by SIEM and aims to gauge its feasibility and future directions.

1.5 Similar Works

One early effort for unstructured log analysis was done by Qiang Fu et al. [1]. In this paper, Fu et al. introduced an algorithm to detect execution anomalies through unstructured logs of Hadoop and SILK. The main difference of this study from ours is that Fu et al. used regular expression to extract specific "log keys" which are predefined based on specific applications. That is, the information extractor already knows what to search 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 the key concept of 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 [2]. In this paper, the authors present the method of using frequent pattern mining and distribution estimation techniques to discover a dominant pattern, and then, use principal component analysis for anomaly detection. An unusual approach of this work is that the authors suggested the analysis of source code to eliminate the uncertainty inherent in parsing application logs. Although this method will give highly accurate results in unstructured log analysis, it is not easily adopted to general cases where the log analyzer does not have access to the source code.

Azodi et al. presents a method to improve IDS/SIEM performance by detecting the input log type and format using regular expression and normalizing log entries [3]. The philosophy behind this approach is very similar to ours. The difference is that their normalized logs are still

fed into rule-based detection, whereas we are exploring the use of machine-learning detection, in a more concerted effort towards a generic parser and detector.

Many other studies related to more specific topics are discussed in the different sections of this report.

Chapter 2. SKAION 2006 IARPA Dataset

2.1 Description

The dataset used for the experiments is from the Packet Clearing House SKAION 2006 IARPA Dataset[1] [4, 5]. 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 [4]. The malicious attacks are of various levels of sophistication ranging from a simple CGI Overflow to attacks involving email phishing [5]. The dataset also includes data related to 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 [4].

The total size of the dataset is 119.2 TB, and a large portion of it contains network traffic traces. For this research, approximately 15 GB of text data from release 4 is used. The data for this study consists mainly of logs collected from 136 sources for different attack scenarios and background traffic. Listings 2-6 are examples of raw logs (IP addresses are replaced with random strings).

```
Thu Sep 22 14:27:41 2005 1 XXX.XXX.XXX.XXX 18 495489
/var/ftp/ftp.sgc.osis.gov/pub/foia/txt/ERKS.pdf b _ o a res@XXX.XXX.XXX.XXX ftp 0 * c
Thu Sep 22 14:28:16 2005 1 XXX.XXX.XXX.XXX 495489
/var/ftp/ftp.sgc.osis.gov/pub/foia/txt/ERKS.pdf b _ o a res@XXX.XXX.XXX.XXX ftp 0 * c
Thu Sep 22 14:28:54 2005 1 XXX.XXX.XXX.XXX 1491
/var/ftp/ftp.sgc.osis.gov/pub/foia/graphics/stars.jpg b _ o a res@XXX.XXX.XXX.XXX ftp 0 * c
```

[Listing 3] FTP Log

[1] 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.

```
Sep 22 15:05:51 www kernel: NETDEV WATCHDOG: eth1: transmit timed out
Sep 22 15:05:51 www kernel: eth1: Transmit timed out, status 0000, PHY status 786d, resetting...
Sep 22 15:06:23 www kernel: NETDEV WATCHDOG: eth1: transmit timed out
Sep 22 15:06:23 www kernel: eth1: Transmit timed out, status 0000, PHY status 786d, resetting...
Sep 22 15:06:27 www kernel: NETDEV WATCHDOG: eth1: transmit timed out
Sep 22 15:06:27 www kernel: eth1: Transmit timed out, status 0000, PHY status 786d, resetting...
Sep 22 15:06:31 www kernel: NETDEV WATCHDOG: eth1: transmit timed out
```

[Listing 4] UNIX Log

```
XXX.XXX.XXX.XXX - - [22/Sep/2005:14:39:41 -0400] "GET / HTTP/1.1" 304 - "-" "Mozilla/4.0
(compatible; MSIE 5.01; Windows NT 5.0)"
XXX.XXX.XXX.XXX - - [22/Sep/2005:14:40:32 -0400] "GET /180.html HTTP/1.1" 200 3558 "-" "TGS Web
Bot"
XXX.XXX.XXX.XXX - - [22/Sep/2005:14:40:32 -0400] "GET /0010.jpg HTTP/1.1" 200 5310 "-" "TGS Web
Bot"
XXX.XXX.XXX.XXX - - [22/Sep/2005:14:45:41 -0400] "GET /100.html HTTP/1.1" 200 3004 "-"
"Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)"
XXX.XXX.XXX.XXX - - [22/Sep/2005:14:45:55 -0400] "GET /1320.html HTTP/1.1" 200 3054 "-" "TGS
Web Bot"
XXX.XXX.XXX.XXX - - [22/Sep/2005:14:45:55 -0400] "GET /0044.jpg HTTP/1.1" 200 4249 "-" "TGS Web
Bot"
```

[Listing 4] Web Access Log

```
8/6/2005       4:57:00 AM     4       0       420     NNTPSVC N/A     HOST23 Pickup    Directory
Status Report.   In the last 60 minutes, the pickup directory for    virtual server 1 has
successfully received 0 articles.
8/6/2005       5:36:26 AM     2       0       8021    BROWSER N/A     HOST23 The   browser   was
unable to retrieve a list of servers from the browser master \\OSIRIS on the network
\Device\NetBT_Tcpip_{78BEE437-352C-477E-9372-546DF7B52119}.  The data is the error code.
8/6/2005       5:57:00 AM     4       0       421     NNTPSVC N/A     HOST23 Post Status Report.
In the last 60 minutes, the virtual server 1   has successfully received 0 posts.
8/6/2005       5:57:00 AM     4       0       420     NNTPSVC N/A     HOST23 Pickup    Directory
Status Report.   In the last 60 minutes, the pickup directory for    virtual server 1 has
successfully received 0 articles.
8/6/2005       6:52:39 AM     2       0       8021    BROWSER N/A     HOST23 The   browser   was
unable to retrieve a list of servers from the browser master \\ HOST23 on the network
\Device\NetBT_Tcpip_{78BEE437-352C-477E-9372-546DF7B52119}.  The data is the error code.
8/6/2005       6:57:00 AM     4       0       421     NNTPSVC N/A     C-HOST3 Post Status Report.
In the last 60 minutes, the virtual server 1   has successfully received 0 posts.
```

[Listing 5] Windows Log

```
[**] [104:3:1] Spade: Non-live dest used: local dest, est. flags: 1.0000 [**]
09/22-14:26:20.871503 XXX.XXX.XXX.XXX:52734 -> YYY.YYY.YYY.YYY:80
TCP TTL:114 TOS:0x0 ID:37695 IpLen:20 DgmLen:40
***A**** Seq: 0x23BFAB1E  Ack: 0xC0FFAB1E  Win: 0x400  TcpLen: 20

[**] [104:3:1] Spade: Non-live dest used: non-err icmp, local dest: 1.0000 [**]
09/22-14:26:26.885387 XXX.XXX.XXX.XXX -> YYY.YYY.YYY.YYY
ICMP TTL:114 TOS:0x0 ID:33547 IpLen:20 DgmLen:28
Type:8  Code:0  ID:63603   Seq:44542  ECHO
```

[Listing 6] Snort Alert

2.2 Attack Scenarios

Release 4 includes data from ten different attack scenarios. Since the logs included for each scenario are not consistent for all scenarios, only the following five shown in Table 1 were included in this study.

Attack ID	Scenario	Background 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 Exfiltration	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 Misconfiguration	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.

[Table 1] Attack Scenarios [4]

Attacks 4s1, 4s3 and 4s4 have a background attack scale of 50%, which means that similar attempts were observed and recorded in background data. This will make it harder to distinguish the alerts and log entries of these three attacks from that of background data. Also, 4s3 involves decoy attacks which produce the same footprints in the intrusion detection system (IDS).

Chapter 3. Text Classification of SKAION Log Messages

3.1 Log Classification and Threat Detection

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 feed them into the right classification algorithm.

There have been many previous studies on text classification of standard natural language corpus [20, 21, 22, 23, 24, 25], scanned OCR documents [28, 29], and social media data [30, 31, 32, 33]. 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 the name of executables, status codes, and error codes. Some are in binary, octal, or hexadecimal number 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 through the 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 for this task and establish a baseline performance.

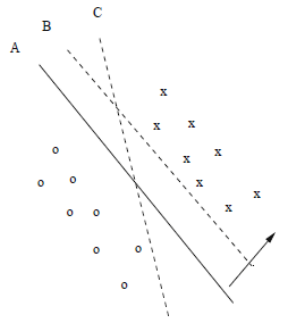
2) Identify features that may be useful for classification and extract those features from the unstructured message. Repeat the same experiments as in (1) using a) the extracted features only, and b) both features and the message together.

3) Analyze the results to determine which features and algorithms perform well in problems of threat detection using unstructured log analytics.

3.2 Classification Algorithms

Since text can be modeled as quantitative data with word frequencies, (we will see how this is done in detail in Section 3.3), a wide variety of classification algorithms developed for numerical or categorical data can also be applied to text classification. However, the high dimensionality and sparsity characteristics of text data makes certain algorithms more suitable for text data [13]. Among the common classification algorithms surveyed by different researchers [13, 14, 15] for text classification, three classification algorithms, the Naïve Bayes Multinomial (NBM), Support Vector Machine (SVM), and Random Forest (RF) generally performed better in terms of accuracy, precision and speed on the SKAION dataset (Section 5.3) The following sections briefly describes the inner workings of these algorithms.

3.2.1 Naïve Bayes Multinomial (NBM)



[Figure 2] Hyperplanes separating two classes [13]

The Naïve Bayes is a probabilistic classifier that models the distribution of the documents in each class using a model based on independence assumptions about the distributions of different terms. Essentially, it computes the posterior probability of a class based on the distribution of the words in the document and ignores the actual position of these words [13]. The Bayes Multinomial Model captures the frequencies of terms in a document and calculates the conditional probability that the document D is from class c_i, using Bayes rule:

$$P(C^D = c_i \mid D = (T,F)) = \frac{P(C^D=c_i)*P(D=(T,F) \mid C^D=c_i)}{P(D=(T,F))} \qquad \text{(Eq. 3.5.1.1)}$$

$$\approxeq P(C^D = c_i) * P(D = (T,F) \mid C^D = c_i),$$

$$where\ T = set\ of\ terms\ in\ D, F = frequencies\ of\ T\ in\ D.$$

and,

$$P(D = (T,F) \mid C^D = c_i) = \frac{|D|!}{\prod_{i=1}^{m} F_i!} * \prod_{t_j \in T} P(t_j \in D \mid C^D = c_i)^{F_j}. \qquad \text{(Eq. 3.5.1.2)}$$

Compared to the multi-variate Bernoulli model, another model in the Naïve Bayes classifier family, the Multinomial model, is known to work well with data of a large vocabulary size [13].

3.2.2 Support Vector Machine (SVM)

The main principle of SVM is to determine the separators in the search space which can best separate the different classes. In Figure 2, there are three hyperplanes separating two classes,

represented by x and o. It is clear that hyperplane A provides the best separation because the normal distance of any of the data points from it is the largest. The separator which represents the maximum margin of separation has the most discriminating power [13]. The advantage of the SVM method for text classification is that it is robust to high dimensionality [15]. Although it is not necessary to use a linear function for an SVM classifier [16], it is very often used in practice for its simplicity [13].

3.2.3 Random Forest (RF)

The Random Forest is an ensemble classifier consisting of a collection of tree-structured base classifiers. Let *D* be a set of documents, and *Nf*, features. The following algorithm builds a Random Tree classifier [17]:

1. Generate k subsets of D { D1, D2, … Dk} by random sampling

2. For each dataset Dk, build a decision tree model by randomly sampling a subspace of m dimension (m < Nf) from the features at each node. Compute all possible splits based on those m features. The data partitions from the best split (e.g. the largest Gini measure) are used to generate child nodes. Repeat until the stopping criterion is reached.

3. Combine k unpruned trees into a Random Forest ensemble and use the majority votes among the trees to reach a classification decision.

The Random Forest ensemble method is known to increase the accuracy of single-decision tree classifiers by returning a classification decision based on decisions from all decision trees [17].

3.3 Text Transformation

In order to classify a log message using machine learning, the log message has to be transformed into a numeric vector that can be used by classification algorithms. Without any specific feature extraction, the message can be transformed into a word vector representation, [6] where each log entry is represented as a vector of bits or integers that indicate whether the message contains a specific word (a bit) or the frequency of occurrence of a word in the message (an integer). In our experiments with the SKAION dataset using the WEKA machine learning tool [8], some common text transformation techniques such as stemming, removing stop words, or using n-gram features have no positive impact on the performance of the classifier with log messages. On the other hand, TF-IDF transformation generally has a positive impact on classifier performance, as long as it is used with appropriate attribute selection strategies. (Figure 3-7)

TF-IDF transformation is a common technique to compute the weighting of words. The TF-IDF score of the word, *j*, in a document, *d*, is calculated with the following formula (Eq. 3.3.1 and Eq. 3.3.2) or some variation [8]. In our case, a document is a log entry and a word is any tokenized string occurring in the set of log entries.

$$TF\_IDF\ (j,d) = TF(j,d) * IDF(j). \qquad \text{(Eq. 3.3.1)}$$

$$IDF(j) = \log\left(\frac{N}{DF(j)}\right). \qquad \text{(Eq. 3.3.2)}$$

where, N is the total number of documents, DF(j)is the number of documents containing the word j, and TF(j, d)is the frequency of the word j in a document d.

In the experiment with SKAION log data, the average precision of the three classifiers improves by 2.3-2.7% if TF-IDF measures are used with attribute selection (Figure 3, message_with_tf-idf_and_ig versus message_no_tf-idf_no_select). A 0.23% increase in precision

is observed when the TF-IDF measure is used with attribute selection based on Information Gain (IG) compared to using attribute selection alone (Figure 3, message_with_tf-idf_and_ig versus message_no_tf-idf_with_ig). The TF-IDF measure's impact on classification performance is also dependent on the classification algorithm. Figure 5-7 shows it has greater impact on the Naïve Bayes Multinomial (NBM) and Random Tree (RT) algorithm than Support Vector Machine (SVM). For NBM, using the TF-IDF measure increased precision and decreased recall (Figure 5, message_with_tf-idf_and_ig versus message_no_tf-idf_with_ig), whereas the opposite effects are observed for RT (Figure 7, message_with_tf-idf_and_ig versus message_no_tf-idf_with_ig). SVM did not show significant changes in performance, based on the TF-IDF transformation (Figure 6).

3.4 Attribute Selection

Attribute Selection is a technique for reducing dimensionality by removing non-informative attributes selectively. Yang and Pederson report in their comparative study, that attribute selections based on Information Gain and the chi-squared test are most effective for text classification [9]. In our experiment with the SKAION dataset unstructured log analysis, both the Information Gain attribute selection and chi-squared test increased the performance and reduced the training and testing time by a similar level (Figures 3, 4).

3.4.1 Information Gain

Information Gain measures the number of bits of information obtained for category prediction by knowing the presence or absence of a term in a document [9, 10] (a log entry in our case). For a classification with m classes ($c_{i\ldots m}$) the Information Gain of term *t* is defined to be [9, 11]:

$$G(t) = -\sum_{i=1}^{m} P(c_i) * \log P\,(c_i)$$

$$+P(t) * \sum_{i=1}^{m} P(c_i|t) * \log P(c_i|t) \\ +P(\bar{t}) * \sum_{i=1}^{m} P(c_i|\bar{t}) * \log P(c_i|\bar{t}).$$ (Eq. 3.4.1)

For each unique term in a set of log messages, we can calculate Information Gain and select only the terms that are above a pre-defined threshold.

3.4.2 Chi-squared Test

The Chi-squared statistic measures the dependency between a feature and the target and can be compared to the chi-square distribution with one degree of freedom to judge extremeness [9, 12]. Let t be the term and $c_i$be the class. Then the Term Goodness measure is defined to be [9]:

$$\chi^2(t, c_i) = \frac{N[P(t,c_i)*P(\overline{t},\overline{c_i}) - P(t,\overline{c_i})*P(\overline{t},c_i)]^2}{P(t)*P(\overline{t})*P(c_i)*P(\overline{c_i})},$$ (Eq. 3.4.2)

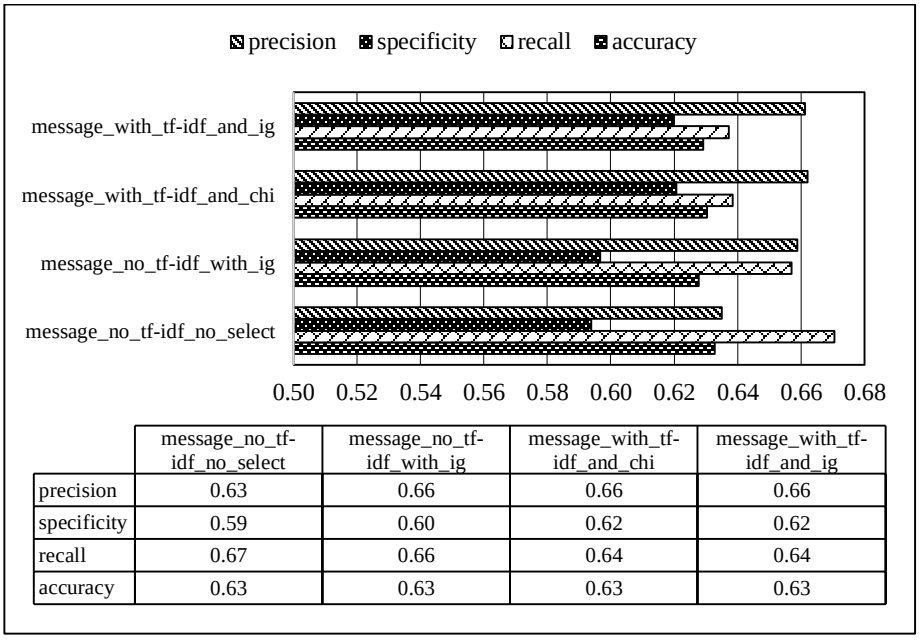
where N is the total number of documents.

This value is zero when t and c_i are independent. To select a feature, the chi-squared value is calculated for each unique term and ranked. Figure 3 and 4 show the relative gain of time and accuracy for a chi-test attribute selection (message_with_tf-idf_and_chi).

3.5 Experimental Results – Text Transformation and Attribute Selection

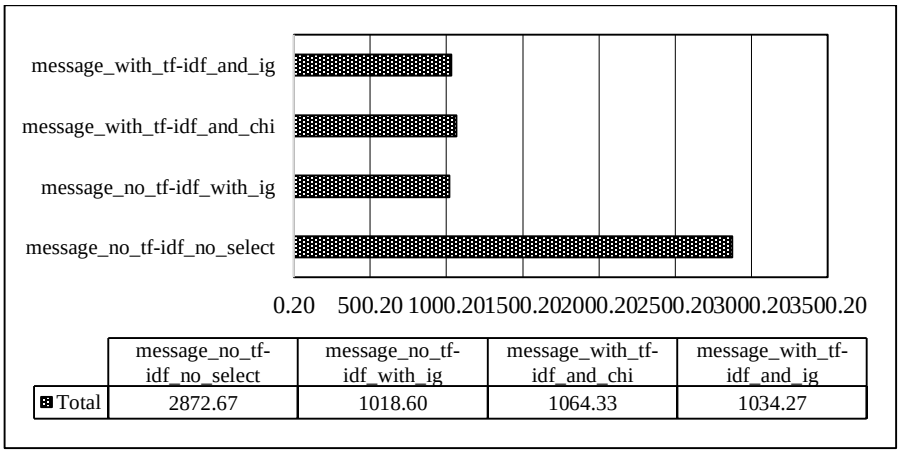
Figures 3-7 are the results of experiments with different combinations of TF-IDF transformations and attribute selections on unstructured log messages from the SKAION dataset. The sample includes 500 log samples from each attack type and the same number of logs from background data. Three classifiers, NBM, SVM, RT, were used in order to gauge the impact to different classifiers.

Figure 3 shows the average performance of all three over five attack types. Overall, the precision increased by 2.5% and recall decreased by 3.0% using TF-IDF measures with attribute selection. A more significant impact on dimensionality reduction was observed in elapsed time for training and testing. As shown in Figure 4, both IG (Information Gain) and the Chi-squared test achieved about a 63% reduction in time. Time was measured by elapsed time for training the model and 10-fold testing.



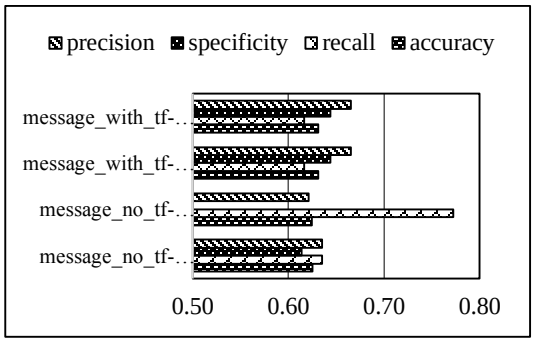
	message_no_tf-idf_no_select	message_no_tf-idf_with_ig	message_with_tf-idf_and_chi	message_with_tf-idf_and_ig
precision	0.63	0.66	0.66	0.66
specificity	0.59	0.60	0.62	0.62
recall	0.67	0.66	0.64	0.64
accuracy	0.63	0.63	0.63	0.63

[Figure 3] Average Performance Based on TF-IDF and Attribute Selection.

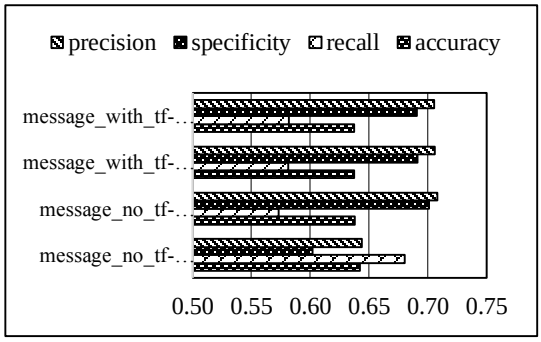


	message_no_tf-idf_no_select	message_no_tf-idf_with_ig	message_with_tf-idf_and_chi	message_with_tf-idf_and_ig
Total	2872.67	1018.60	1064.33	1034.27

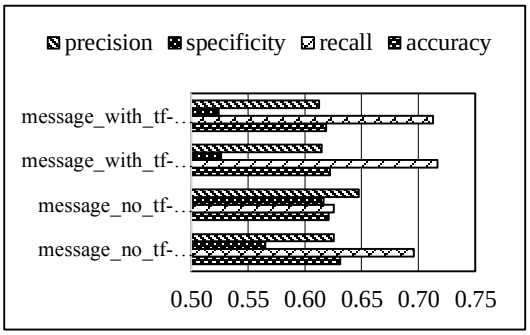
[Figure 4] Average Time in ms for Training and testing (10 folds)



[Figure 5] Average Performance - NBM



[Figure 6] Average Performance - SVM



[Figure 7] Average Performance - RT

Chapter 4. Feature Extraction

4.1 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 detection of security threats. Conversely, collection time (the time at which the log entry is collected by the agent) is usually used as a secondary time reference when generation time (time at which the log entry is generated at the source) is not available.

The SKAION data is collected off-line using scripts and, thus includes little meta-data. During aggregation and sampling of the log entries, the following meta-data (Table 2) are kept for reference purposes. Only two of them: log source and message length, were used for classification.

Feature	Source	Used for Classification
file_name	Full path of the raw log file	No
line_number	Line number of the log entry in the raw log file	No
message_length	Number of characters in message in its original format	Yes
log_source	Source device name concatenated by system, o/s or application name	Yes

[Table 2] SAKION Log Meta-data

4.2 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 [18, 19]. In our context, we are interested in recognizing the things that can be found in log messages 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 would be good indicators for log correlation and threat detection. 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 3 describes the entities 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

[Table 3] Entities of Interest

The CRF classifier from the Stanford NLP library [19] 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 4 shows the average performance per entity category.

Using a Conditional Random Field (CRF)-based statistical NER system to extract entities has certain advantages over using pattern matching through regular expression. The CRF classifier models the sequence of words, rather than individual words separately. Therefore, it recognizes

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
Weighted Avg.	0.9886	0.0055	0.9886	0.9886	0.9886

[Table 4] Feature Extraction Performance using Stanford NER

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 training data.

For example, if the CRF classifier is trained to recognize the grey parts in Listing 7 as application names, then with high chance it also recognize "SNMP event log extension agent" as an application name in Listing 8 even if the log entry was not included in the training data.

```
The Windows Media Unicast Service started

The database engine 6.00.3940.0013 started

The File Server for Macintosh service was unable to contact a
domain controller

WMI ADAP was unable to process the PerfDisk performance library
```

[Listing 7] CRF Classifier Training Log Entries

```
SNMP event log extension agent is starting
```

[Listing 8] CRF Classifier Test Log Entry

4.3 Log Type Classification

Log type is a generic classification of log entry that indicates the purpose or logging level, such as information, error, audit, warning, etc. This can be useful information for threat detection. For example, we can have a hypothesis that if the number of error logs or alerts increases significantly, then the system is not in a normal state. Two problems were found in the SKAION dataset to determine log types reliably. First, some free text logs did not have this information at all, whereas some structured logs contain log type as meta-data, such as Windows Event Type, or Syslog Priority. Second, there is no standard way of labeling a log entry as one of the log types. For example, Windows Event Type 1180 cannot be normalized as a category that is similar to a

Syslog Priority level, error. Therefore, we cannot reliably interpret the real importance, or priority based on the meta-data.

To overcome these problems, this study determines the log types from the log message using an SVM classifier. 200 log entries were sampled from each of the different class (five attack data and one background data) and manually labeled. Using the WEKA machine learning library [8], the SVM model is trained. In a standard 10-fold test, an average of 98.5% log messages were labeled correctly. (See Table 5)

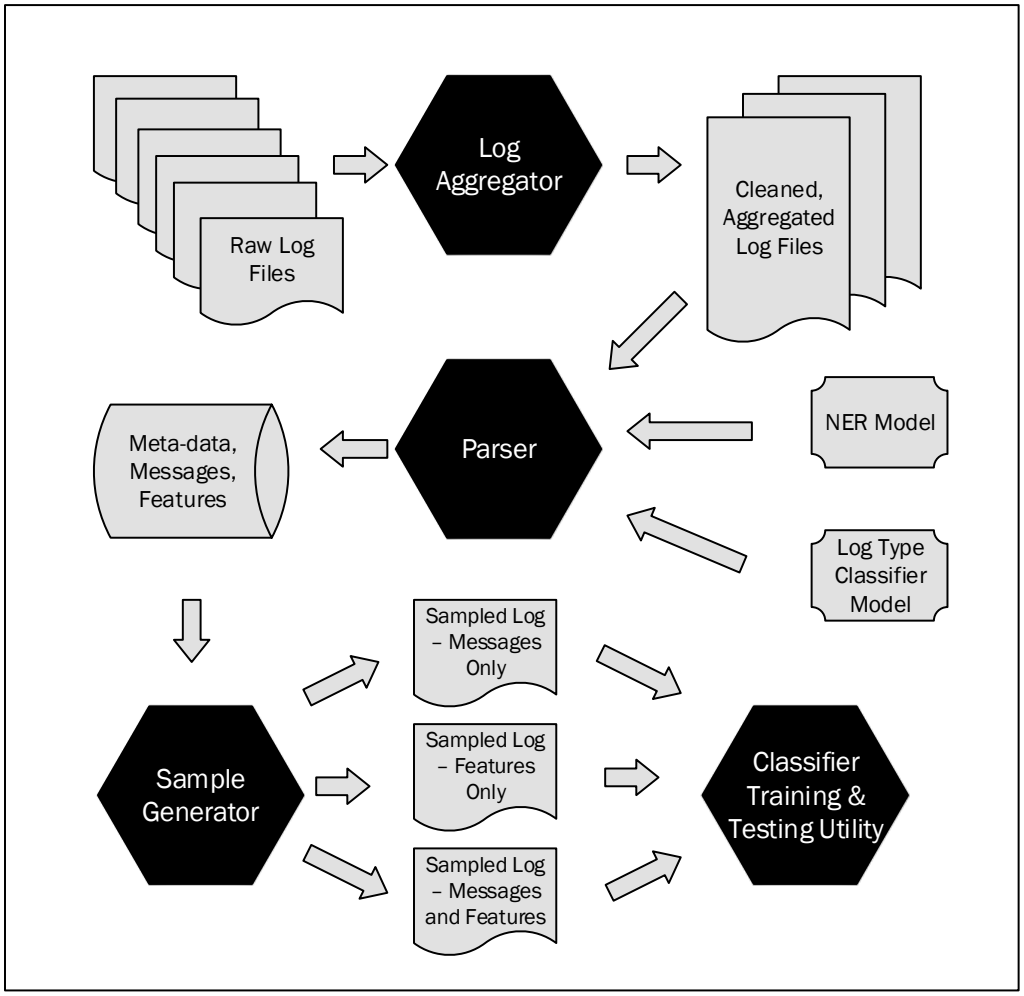
Log Type Class	TP Rate	FP Rate	Precision	Recall	F Measure
AUDIT	0.998	0.013	0.986	0.998	0.992
INFO	0.971	0.007	0.966	0.971	0.968
ERROR	0.833	0.003	0.938	0.833	0.882
WARN	1	0	1	1	1
ALERT	0.984	0	1	0.984	0.992
Weighted Avg.	0.985	0.007	0.985	0.985	0.985

[Table 5] Log Type Classification

Chapter 5. Performance Analysis

5.1 Tools, Libraries, and Programs

A custom-developed application written in Java, Generic Log Analyzer (GLA) is used to perform all experiments described in this section. Figure 8 summarizes the data flow and components of GLA, showing how the raw log files are processed, parsed, and sampled for the training and testing of classifiers.



[Figure 8] Log Aggregation, Sampling, & Classification by GLA

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[2] (Command-Line Utilities, and Java-API)
- WEKA v.3.6.13[3] (GUI, Command-Line Utilities, and Java-API)
- R v. 3.2.0[4]
- Rattle v.3.4.1[5]

5.2 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 without any successful attack. 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 a log type classifier. The models for the classifiers are trained separately, as described in Chapter 4.

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 background data. For example, a data file of sample size 500 contains 500 log samples from one of the attack types

[2]Licensed under the GNU General Public License (v2 or later)
[3] © 1999-2015 The University of Waikato, Hamilton, New Zealand
[4] © 2015 The R Foundation for Statistical Computing
[5] © 2006-2014 Togaware Pty Ltd.

and 500 log samples from background traffic data. Table 6 shows the proportion of the sample size to the population size.

Attack ID	Population Size	Sample Size			
		500	1000	1500	2000
4s1	115242	0.43%	0.87%	1.30%	1.74%
4s3	112532	0.44%	0.89%	1.33%	1.78%
4s4	178895	0.28%	0.56%	0.84%	1.12%
4s13	146163	0.34%	0.68%	1.03%	1.37%
4s14	174450	0.29%	0.57%	0.86%	1.15%
b2 (background – no attacks)	149284	0.33%	0.67%	1.00%	1.34%

[Table 6] Sample Size Proportion per Attack ID

The sampled logs are further processed into three different data sets:

1. **Message-only data**: contains a labeled free-text message from the log file. The message is stripped of any string indicating time or date 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. However, we do not want the classifier model to fit to time/date strings.

2. **Features-only data**: contains the extracted features (user, application, host keywords) and meta-data (message length, source system) without any original free-text message. All features indicating time or file locations are removed for classification.

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 for classification.

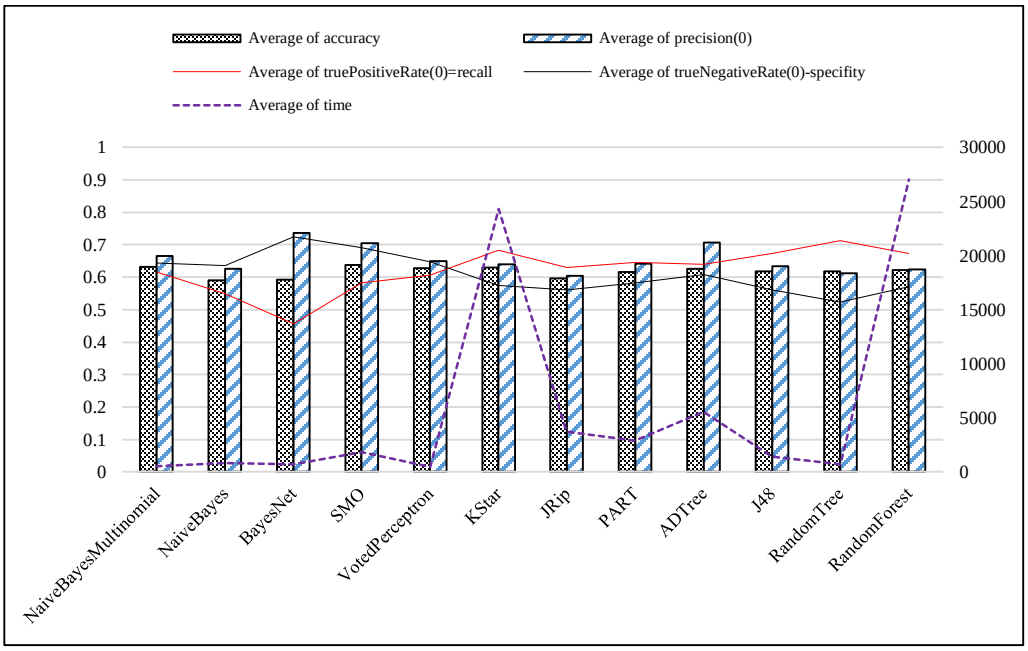
5.3 Classifier Performance

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 [8] over 3 different types of datasets: message-only, features-only, and both message and features. To evaluate performance, the following five statistics were used:

- 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. On the other hand, accuracy and precision are positively related. For our evaluation, we preferred classifiers with little variance among the four performance indicators, since larger variance means that a good performance in one of these measures may lead to a larger error in the other measure. Time represents the relative complexity of the classification algorithm and does not show any relation to other performance indicators in terms of evaluation of the classification algorithm. Therefore, a complex classification algorithm that takes longer to build and classify does not necessarily perform better.

Figure 9 and Table 7 show that the Naïve Bayes Multinomial (NBM), Voted Perceptron (VP), and AD Tree (ADT) algorithms perform best for message-only data. These three algorithms display high values of performance statistics with little variances among them. As expected, the recall rate (red line) and specificity (black line) show a reverse correlation, while time (purple dotted line) does not show any relation to performance.

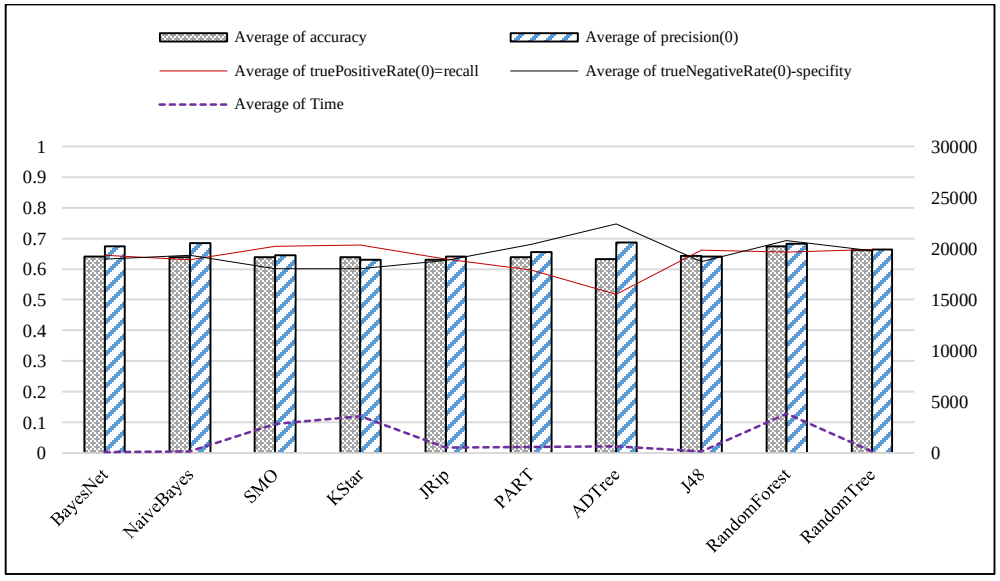


[Figure 9] Classifier Performance – Message-only

Classifier	Time (ms)	Rank	Avg. Perf	Rank	StdDev	Rank
BayesNet	725.4	4	0.62718	6	0.13091	12
NaiveBayes	813.6	5	0.60039	11	0.03925	5
NaiveBayesMultinomial	**543.8**	**2**	**0.63933**	**3**	**0.02054**	**2**
SMO (SVM)	1862.2	7	0.65392	1	0.05622	10
VotedPerceptron	**491.2**	**1**	**0.63341**	**4**	**0.02040**	**1**
KStar	24293.2	11	0.63150	5	0.04439	8
JRip	3704.8	9	0.59777	12	0.02825	3
PART	2891.0	8	0.62200	9	0.02898	4
ADTree	**5567.0**	**10**	**0.64516**	**2**	**0.04379**	**7**
J48	1430.8	6	0.62205	8	0.04701	9
RandomForest	27013.4	12	0.62290	7	0.04183	6
RandomTree	718.8	3	0.61729	10	0.07722	11
Max	27013.4		0.65392		0.13091	
Min	491.2		0.59777		0.02040	

[Table 7] Performance Ranking – Message-only data

For features-only data, (Figure 10) Bayes Net (BN), Random Forest (RF) and Random Tree (RT) show strong performance. Average time for training and testing is significantly less when we use feature only data than when data is classified with free-text messages. Free-text messages generally yield a large number of feature sets with counts of 120-160 after attribute selection in 1000 sample entries, whereas features-only data has 12-18 attributes after the selection. Both the NBM and VP algorithms which performed well for message-only data, can handle only numerical data and therefore, was not tested for features-only or message-and-features data, as they contain a large amount of categorical data.

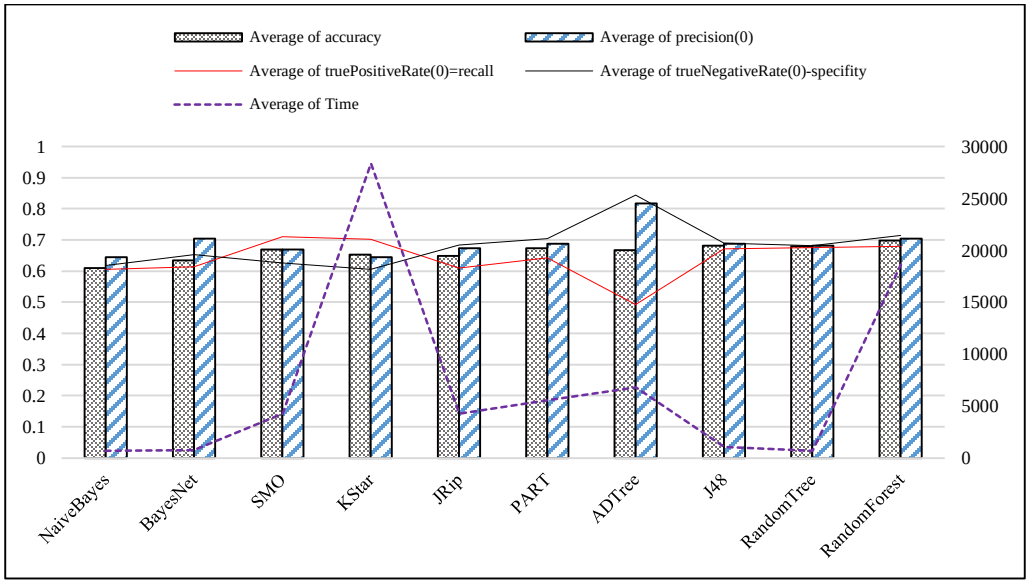


[Figure 5] Classifier Performance - Features Only Data

	Time (ms)	Rank	Avg. Perf	Rank	StdDev	Rank
BayesNet	**107.2**	**1**	**0.64849**	**4**	**0.01826**	**5**
NaiveBayes	126.2	2	0.65033	3	0.02408	6
SMO (SVM)	2822.4	5	0.63909	8	0.02987	7
KStar	3612.2	10	0.63727	9	0.03228	8
JRip	543.2	6	0.63344	10	0.00580	2
PART	583.4	7	0.64291	6	0.03508	9
ADTree	629.8	8	0.64610	5	0.09681	10
J48	126	4	0.64247	7	0.01552	4
RandomForest	**3870.6**	**9**	**0.67660**	**1**	**0.01549**	**3**
RandomTree	**127.8**	**1**	**0.66212**	**2**	**0.00130**	**1**
Max	3870.6		0.67660		0.09681	
Min	107.2		0.63344		0.00130	

[Table 8] Performance Ranking – Features-only

For data with both messages and extracted features (Figure 11 and Table 9), tree-based algorithms such as Random Tree (RT), J48 Tree (J48), and Random Forest (RF), performed better than others. Generally, classifiers performed best on this type of dataset. As shown in Table 9, the maximum performance reached 0.7054 for this dataset, compared to 0.6766 and 0. 6539 for feature-only and message-only datasets, respectably. On the other hand, the classification of these data sets took an average of 19.6% longer than classifying message-only data and 140% longer than features-only data.



[Figure 6] Classifier Performance – Message-and-Features

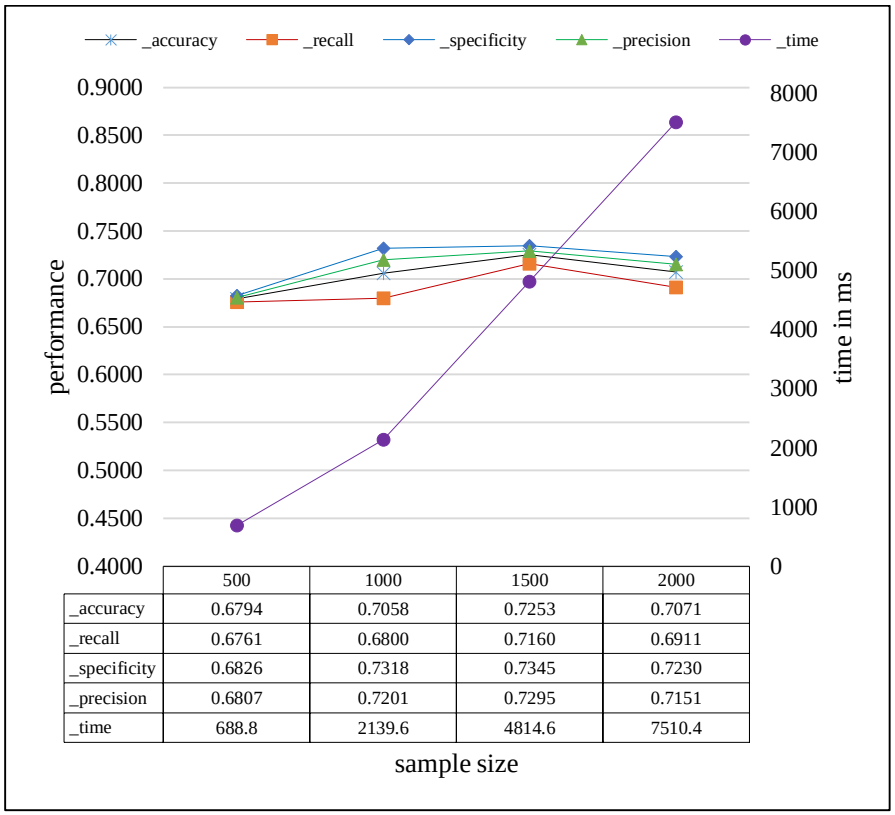
	Avg Time (ms)	Rank	Avg. Perf	Rank	StdDev	Rank
BayesNet	782.2	3	0.65171	8	0.03867	8
NaiveBayes	694.2	2	0.61956	10	0.01812	4
SMO (SVM)	4263.0	5	0.66904	6	0.03409	7
KStar	28340.2	10	0.65139	9	0.04034	9
JRip	4290.2	6	0.65369	7	0.03319	6
PART	5553.4	7	0.67764	5	0.02622	5
ADTree	6819.0	8	0.70541	1	0.16163	10
J48	**1086.2**	**4**	**0.68278**	**3**	**0.00805**	**2**
RandomForest	**18604.0**	**9**	**0.69942**	**2**	**0.01480**	**3**
RandomTree	**678.8**	**1**	**0.67972**	**4**	**0.00274**	**1**
Max	28340.2		0.70541		0.16163	
Min	678.8		0.61956		0.00274	

[Table 9] Performance Ranking – Message-and-Features

5.4 Performance by Sample Size

If the size of training data increases, would the classification performance increase as well? To answer this question, we ran the classification with four different sample sizes: 500, 1000, 1500, and 2000. It should be noted that the sample size refers to the number of log entries from each attack type, and the same number from background data. That means the 4s1 attack type data with sample size 500 contains 1000 log samples, half of which is from 4s1 and the other half from b1 (background data). Experimental data contained both message and extracted features. The Random Tree classifier was used with an Information Gain attribute selection of threshold value 0 (only features with a positive IG were selected).

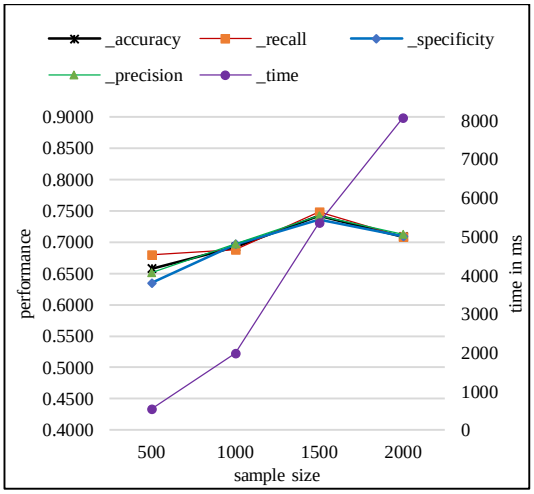
Figure 12 displays the average performance for all five attack types. Generally, the performance statistics increase from 500 to 1500, but fall at 2000 in average data. The reason for



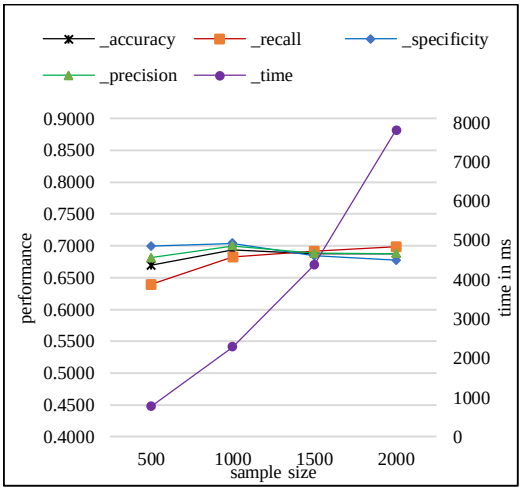
	500	1000	1500	2000
_accuracy	0.6794	0.7058	0.7253	0.7071
_recall	0.6761	0.6800	0.7160	0.6911
_specificity	0.6826	0.7318	0.7345	0.7230
_precision	0.6807	0.7201	0.7295	0.7151
_time	688.8	2139.6	4814.6	7510.4

[Figure 7] Performance by Sample Size, Avg. of 5 attack types

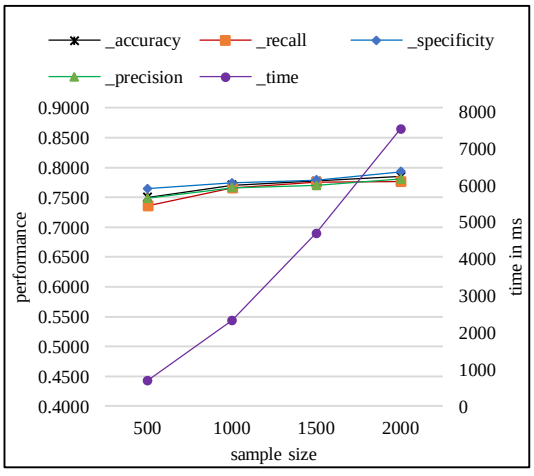
the degradation at 2000 is not clear. The cause could be that the classifiers reached their maximum predictive power on a given dataset between 1500 and 2000, or that the WEKA software reached its maximum capacity in handling larger data. Time, shown in the purple line, seems to increase exponentially. The four performance measures show different trends for each attack type, as shown in Figures 13-18. The general trend of decrease at 2000 is consistent, except in 4s4, which increased slightly at 2000 (Figure 15). The variance among the performance measures decrease as sample size increases with the exception of 4s14. 4s14 shows the lowest level of variance at 500 and fluctuations as the sample size increases (Figure 17).



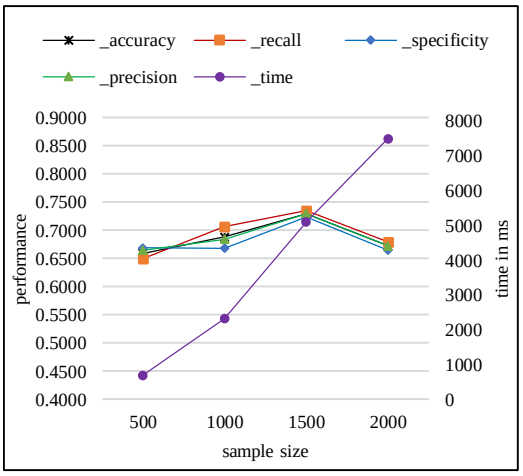
[Figure13] 4s1



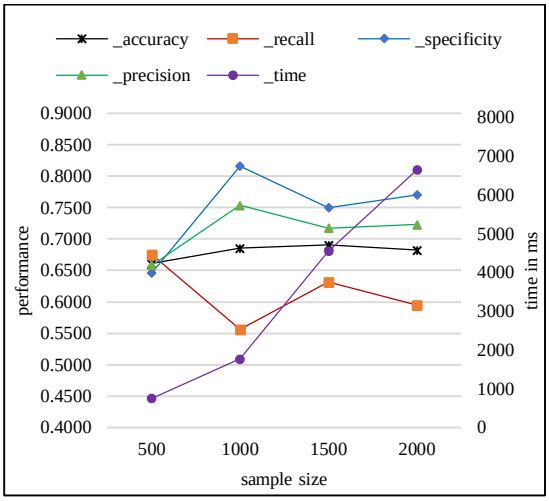
[Figure 14] 4s3



[Figure 15] 4s4



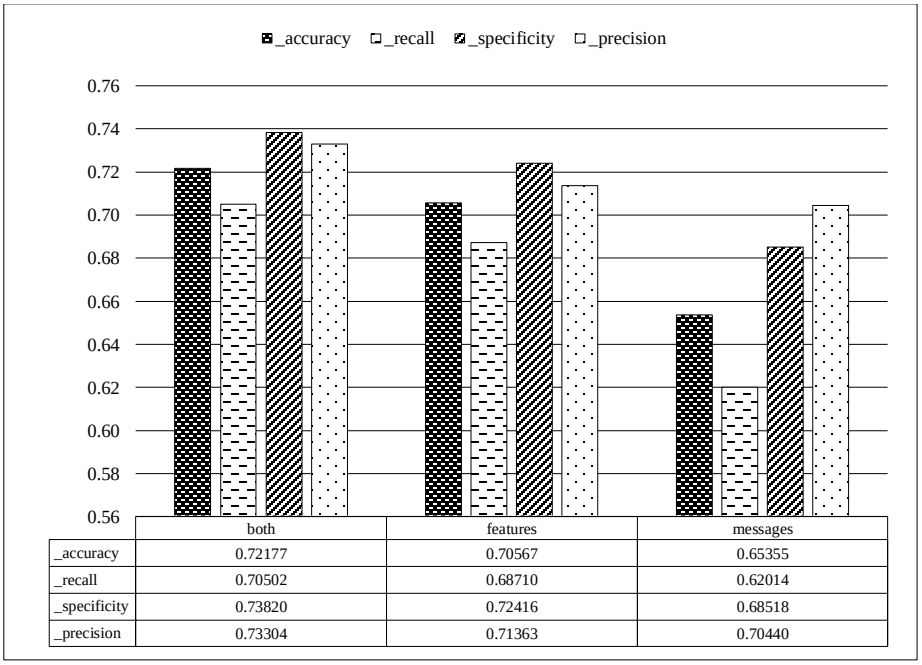
[Figure 16] 4s13



[Figure 17] 4s14

5.5 Performance Gain by Feature Extraction

Feature extraction is the process through which we can attain more obscure information from unstructured logs. With the techniques described in Chapter 4, we extracted relevant information from free-text messages, such as the application name, host, user name, time, and keywords (Section 4.2). We also determined if the log message was an audit log, error log,

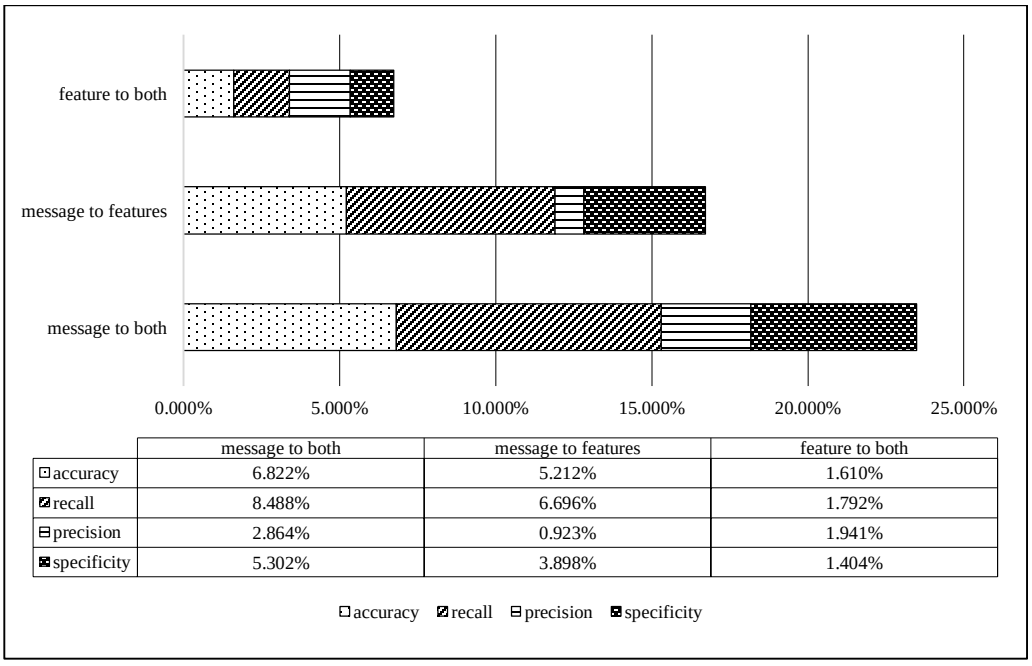


	both	features	messages
_accuracy	0.72177	0.70567	0.65355
_recall	0.70502	0.68710	0.62014
_specificity	0.73820	0.72416	0.68518
_precision	0.73304	0.71363	0.70440

[Figure 18] Performance by Data Type

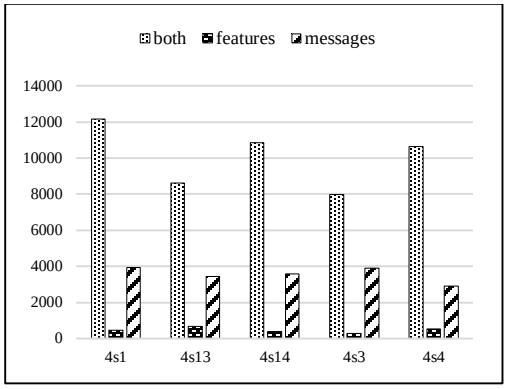
information, warning or alert (Section 4.3). Using these features with the unstructured message generally seemed to increase the classification performance as shown in Section 5.2 and 5.3. In this section, we will more closely examine the performance gain achieved by feature extraction.

In order to compare the best performers among different data types, we selected only the classifiers that performed best for specific data types: NBM and VT for message-only data, RT and RF for feature-only data, and J48 and RT for the data with both message and features. We used a sample size of 1500 which gave the best measures in previous experiments (Section 5.4). For all data types, we used TF-IDF measures and IG attribute selection as it is shown to improve performance (Sections 3.2 and 3.3). Figure 18 shows the average performance of the two classifiers for each data type. As expected, using both a message and features still achieved the highest rate of accuracy, recall, specificity and precision. Figure 19 details the differences between pairs of data types. Compared with message-only data, data with both a message and features gained an

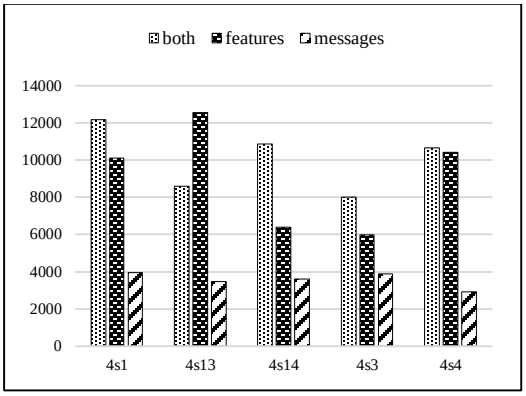


	message to both	message to features	feature to both
accuracy	6.822%	5.212%	1.610%
recall	8.488%	6.696%	1.792%
precision	2.864%	0.923%	1.941%
specificity	5.302%	3.898%	1.404%

[Figure 19] Performance Gain by Feature Extraction



[Figure 20] Avg. Time by Data Type - RF excluded

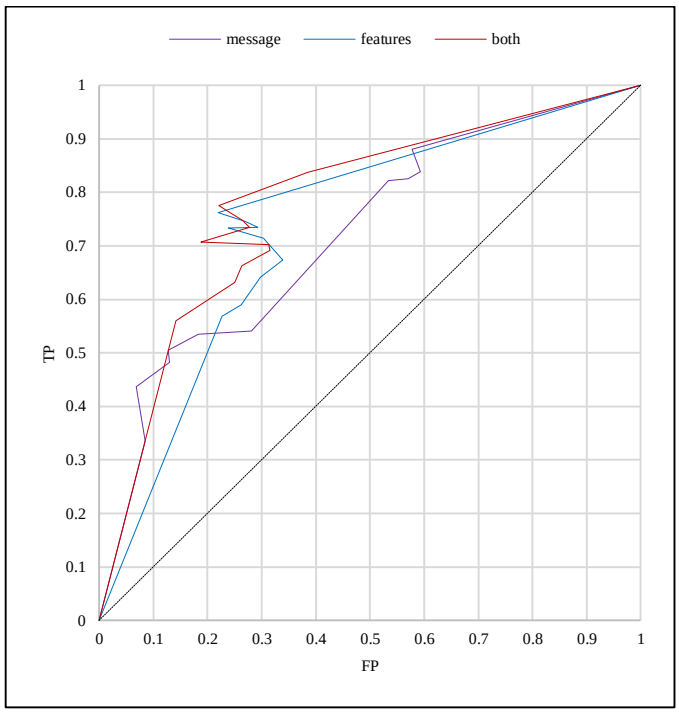


[Figure 21] Avg. Time by Data Type

average of 6.82% accuracy, 8.49% recall, 2.86% precision, and 5.30% specificity. This is a 23.5% cumulative gain. However, this gain in performance came with the cost of time. As shown in Figure 20, the data set with both features and messages took approximately 95% longer to build a model and perform a 10-fold test with 3000 log entries. The values for features-only data in Figure 21 is biased because one of the algorithms used for the data type was RF, which usually takes about 10 times longer than other algorithms when used on other data types. Figure 20 shows the distribution excluding the time for RF, which is more comparable to other data types. The features-only data contains only 19 to 33 attributes, allowing it to run much faster than other data types with which the number of attributes range from 261 to 320. As shown in Figure 20, features-only data shows a large improvement from message-only data and takes only a fraction of the time. This is an important characteristic to consider for real-time analytics.

In the Receiver Operating Characteristic (ROC) curves in Figure 22, the curve for the data type with both messages and features (red) exhibits the largest area under the curve, signifying a

stronger predictive power. The features-only data type (blue) is next, while the message-only data type (purple) has the smallest ROC area.



[Figure 22] ROC Curve by Data Types

Chapter 6. Summary and Discussions

6.1 Summary

From the experiments described in Chapter 3, 4 and 5 we know the following factors affect classification performance for unstructured log analysis:

1) A 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. (Section 5.3)

2) Generally, the size of the training data is positively related to the classification performance, until a peak is reached. This peak observed in the SKAION dataset is between sample sizes of 1500 and 2000 for binary classification. (Section 5.4)

3) Identifying, extracting, and using features such as the application name, host name, IP addresses, user name, and keywords from the unstructured log messages increased the classification performance by 2.8-8.5%. (Section 5.5)

4) Using extracted features only, we can improve the classification performance by 0.93-5.21%, and decrease time by 87%. These are useful characteristics in real-time analytics (Section 5.5)

5) An attribute selection algorithm based on Information Gain or the chi-squared test increased the average precision by 2.5% but decreased the average recall by 3.0%. This also reduced time for training and testing by 63%. (Section 3.3)

6) TF-IDF transformation of the free-text messages has a small but positive impact if it is used with the attribute selection algorithm. (Section 3.2)

6.2 Discussion

Through this study, we discovered many factors that affect unstructured log analysis using machine learning. Still, the results raise some questions: Is a 70-73% predictive performance sufficient for security threat detection? Is a 2.8-8.5% performance gain worth the time and effort of feature extraction? Is the performance peak with the training data size shown in Section 5.4 artificial or natural? We will discuss these questions in detail in this section.

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

It is difficult to find a reliable SIEM benchmark for security threat detection, primarily because SIEM is primarily used as a log collection and archiving tool as stated in Chapter 1. The most common benchmarking metric for SIEM is Event Per Second (EPS), which indicates how many logs the system can handle per second [38]. Since the availability of the KDD99 dataset in 1999 [35], more benchmarking studies have been done on network-based security detection through the Intrusion Detection/Prevention System (IDS/IPS). [34, 36, 37]. Lippmann et al. found 18% of attacks were completely missed by a signature-based Network-based Intrusion Detection System (NIDS) [34]. A more recent study with the same dataset reports detection rates ranging from 5.4 to 99.4%, suggesting network-based IDS performs well for certain security threats, but had little success on others [38]. 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, SIEM can handle information from almost any system, application, or connected device. Considering we achieved a 70-73% prediction using unchartered information within free-text data for a very diverse range of systems, it would not be a large stretch to assume that this approach has even more potential. With further studies on intelligent correlation with the NIDS alerts, email filter logs, or a configuration management system, the overall detection performance could improve significantly.

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

Many statistical analytics experts in the marketing area agree that using non-traditional data sources such as social networks improved their predictions by 0.5-1.5% [39]. When analytics have reached their maximum predictive potential with the existing data and techniques, adding more data to gain another 1 or 2% could be a potential a competitive edge. 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 gain in performance metrics:

a. As described in (4) in Section 6.1, using features alone, we have better prediction than message-only data and decreased the time by 87%. This opens up the possibility of real-time unstructured log analysis using classifiers with a continuous learning capability.

b. The extracted features are important keys for the normalization of log messages and the analysis of the relationships between log entries. This may allow us to further improve the detection rate.

3) Is the performance peak with the training data size shown in Section 5.4 a natural limit for data?

In Section 5.4, we observed that a peak was reached at 1500 for most attack types. If this is the natural limit for this type of data, adding more training data would not improve performance. However, if this limitation is imposed by the software or hardware, then by using infrastructure specialized for Big Data analytics, we may further improve the classification performance. Further work is necessary to answer this question, because there currently seem to be no general rules or guidelines on this subject. The relationship between training data size and performance appears to be highly dependent on the specific dataset being studied [7, 26].

In general, performance analysis indicates a good potential for further studies. The performance test results as well as the techniques described in this study should be useful information to design more sophisticated feature extraction methodologies and algorithms that are specialized in log classification for threat detection.

Chapter 7. Conclusions

This study systematically explores the possibility of utilizing 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 message, 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 along with TF-IDF transformation and IG attribute selection. Using features only, we also achieved a similar 68-71% performance metric, but 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 in SIEM data. Moreover, the methods used this this study produces encouraging results for further studies on automatic log normalization, intelligent feature extraction and entity relationship analysis using Big Data analytics.

References

[1] 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.

[2] 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.

[3] 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.

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

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

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

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

[8] 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.

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

[10] MOULINIER, I., RASKINIS, G. AND GANASCIA, J. 1996. Text categorization: a symbolic approach. In *proceedings of the fifth annual symposium on document analysis and information retrieval*, 87-99.

[11] LEWIS, D.D. AND RINGUETTE, M. 1994. A comparison of two learning algorithms for text categorization. In *Third annual symposium on document analysis and information retrieval*, 81-93.

[12] ZHENG, Z., WU, X. AND SRIHARI, R. 2004. Feature selection for text categorization on imbalanced data. *ACM Sigkdd Explorations Newsletter* 6, 80-89.

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

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

[15] 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*

[16] AIZERMAN, A., BRAVERMAN, E.M. AND ROZONER, L. 1964. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control* 25, 821-837.

[17] WU, Q., YE, Y., ZHANG, H., NG, M.K. AND HO, S. 2014. ForesTexter: an efficient random forest algorithm for imbalanced text categorization. *Knowledge-Based Systems* 67, 105-116.

[18] FINKEL, J.R., GRENAGER, T. AND MANNING, C. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics,* Association for Computational Linguistics, 363-370.

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

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

[21] 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.

[22] 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.

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

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

[25] 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.

[26] 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.

[27] 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.

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

[29] TAGHVA, K., BORSACK, J. AND CONDIT, A. 1996. Effects of OCR errors on ranking and feedback using the vector space model. *Information processing & management* 32, 317-327.

[30] 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.

[31] 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.

[32] 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.

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

[34] 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.

[35] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[36] PFAHRINGER, B. 2000. Winning the KDD99 classification cup: bagged boosting. *ACM SIGKDD Explorations Newsletter* 1, 65-66.

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

[38] 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*.

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

Curriculum Vitae

Candace S. Suh-Lee

Department of Computer Science

Howard R. Hughes College of Engineering

University of Nevada, Las Vegas

Tel: (702) 912-8238

Email: suhlee@unlv.nevada.edu

Research Interest

Cybersecurity, Big Data Analytics, Security Information and Event Analysis, Security Metrics, Network Security, Natural Language Processing, Predictive Analytics

Education

- M. Sc., Computer Science, University of Nevada, Las Vegas, (Expected Spring 2016)

  Thesis topic: Mining Unstructured Log Messages for Security Threat Detection

  Advisor: Dr. Yoohwan Kim, Dr. Ju-yeon Jo
- Hon. B.Sc., Computer Science, University of Toronto, Canada, 2002

  Graduate with High-Distinction, University of Toronto Scholar, 2000, 2001, 2002

Publications

- Suh-Lee, Candace, and Ju-yeon Jo. "Quantifying security risk by measuring network risk conditions." Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference on. IEEE, 2015.

Academic Positions

- **Research Assistant, Department of Computer Science, University of Nevada, Las Vegas, 2014-Current**

  Conduct research under the guidance of the faculty advisor on following areas:

  Security Information and Event Analysis Using Big Data Analytics - the increase of system complexity, data size, and processing power caused an explosive increase of amount of system-generated log messages. Although the messages contains many useful information for security threat detection, the complexity and volume make it very difficult for even most trained analysts to find the right information. While the conventional security event management tools fall short in terms of utility and cost-effectiveness, this research explores a new paradigm in security event analysis using natural language processing, statistical pattern recognition, machine learning and Big Data analytics. The aim is to increase accuracy, sensitivity and specificity of the security tools and to make log analysis fast enough for real-time processing.

  Security Risk Quantification - no more "High/Medium/Low." After decades of intensive investment and specialization, it is about the time that we have the meaningful and concise numerical representation of the security risk. This research aims to come up with a set of numbers that objectively represent various aspects of information security risk arising from

the underlying technologies. Much like ROI or ROE in the financial accounting, these security indices can be tracked, compared, and analyzed providing essential information in decision making.

Awards/Scholarship

- 2nd Place, Poster Science and Engineering, Graduate and Professional Student Association Research Forum, University of Nevada, Las Vegas, 2015
- GPSA Graduate Research Sponsorship Award, University of Nevada, Las Vegas, 2015
- University of Toronto Scholar's Award, 2000, 2001, 2002

Posters/Presentations

- *"Risk Prioritization of Network Vulnerabilities"* Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference, Las Vegas, 2015
- *"Quantifying security risk by measuring network risk conditions"* GPSA Annual Research Forum, University of Nevada, Las Vegas, 2015
- *"Attack Simulation based on Network Vulnerability Risk"* Howard R. Hughes College of Engineering Graduate Poster Competition, University of Nevada, Las Vegas, 2015

Industrial/Consulting Positions

- **Principal Consultant, CSL Security Consulting Co., 2014-Current**

  Developed a series of security questionnaires (740+ questions) for all levels of workforce, in order to evaluate the efforts required to migrate to NERC CIP v.5 standards for a large power generation client; Developed and documented IT Security roadmap for three year horizon, discussing the risk management, IT governance, required technical controls, and

capital/operating cost projection; Established Information Security Programs by selecting security frameworks (ISO/IEC 27001/2), developing policies, standards, and procedures, and obtaining approvals from the senior management

- **IT Security Manager, Liberty Algonquin Business Services, 2013-2014**

  Developed and documented IT Security roadmap for three year horizon, focusing on technology risk management and capital/operating cost projection; Presented IT Security roadmap to IT Steering Committee, securing 100% budget approval and senior management support; Established 3-rd party 24/7 monitoring of network security events with 3rd party SOC; Developed a comprehensive corporate-wide information security program based on ISO27001/2 standards; Supported annual financial and SOX audit by interfacing with external auditors and internal business units; Co-drafted the privacy policy compliant with the different privacy regulations of 9 U.S. states and provinces.

- **Manager, Network and Security Services, AESI Acumen Engineering Solutions International Inc., 2011-2013**

  Developed Smart Grid Cyber Security Master Plan enabling the client to win the DoE's Smart Grid Stimulus Grant; executed the plan by implementing network segregation, SIEM, AAA, Firewall and PKI. Conducted security risk assessment, vulnerability assessments, and security reviews for all new devices and applications commissioned for the Smart Grid; Developed on-going security programs for the risk management, identity and access management, patch management, vulnerability management, and security incident handling; Conducted Security Gap Assessment and Benchmarking Analysis for a large power utility company; Conducted SCADA security assessment for a large power company and benchmarked their security posture against 14 comparable entities. Conducted gap

analysis and program maturity assessment based on Electricity Sector Cyber Security Capability Maturity Model issued by DoE; Collaborated in development of risk-based assessment methodology for identifying critical assets and critical cyber asset for Montana-Alberta 230Kv Tie-Line project; Developed an ICS (Industrial Control System) Security Framework for the city transportation commission operating subway, bus, and light railway system

- **Senior IT Security Specialist, Hydro One Networks Inc., 2008-2011**

  Provided consulting to various business units in developing and maintaining the procedural and technical controls for NERC CIP, Bill 198, ISO/IEC 27001/2 and NIST 800; Initiated and ran security programs for information/system security, regulatory compliance, risk management, and business continuity; Conducted periodic/ad-hoc IT security reviews and reported results with metrics and recommendations; Provided project management support for various security projects; Implemented a Compliance Management System for regulatory compliance for transmission and distribution operation.

- **Systems Engineer/Officer – Transmission Operating Tools, Hydro One Networks Inc., 2004-2008**

  Provided day-to-day support for main computer system for power grid operation; Responded to trouble calls and provided 2nd-level support for SCADA, ICCP, and FEP components; Participated as the SCADA SME in Cyber Security Compliance Project, EMS Software Upgrade, Control Room Hardware Refresh, and Backup Control Center Activation.

- **Assistant Systems Engineer/Officer - Operations Tools & Facilities, Hydro One Networks Inc., 2002-2004**

Developed real-time power system applications to increase the productivity and operational accuracy in the Control Room, using Java, C++, J2EE, JMS and JRE technologies (i.e. Automatic Capacitor Switching, Multiple Tap Changer Control, ICCP Hold-offs, Historical Data Web-reporting, One-click Control Room Information System); Enabled real-time data exchange with 25+ market participants (IESO, OPG, NYPA, and other electricity distributors), by initiating, implementing, supporting, and troubleshooting ICCP communication channels; Enhanced system availability for the in-house developed power system applications, by creating configurable, transparent, automatic logging and security frameworks, using Java, C++, JRE, JMS, JAAS, and AOP technologies

Professional Certifications/Memberships

- Member of American Gas Association Cyber Security Task Force 2013
- Certified Information System Auditor (CISA) 2012
- Certified Information System Security Professional (CISSP) 2009
- GIAC Security Essentials Certification (GSEC) 2009

Interests

- Safe and Secure Online: Volunteering to help children learn how to protect themselves online
- Golf, Hiking, Kendo