Exploring Syntactical Features for Anomaly Detection in Application Logs

Rafael Copstein, Egil Karlsen, Jeff Schwartzentruber, Nur Zincir-Heywood, Malcolm Heywood

Abstract: In this research, we analyze the effect of lightweight syntactical feature extraction techniques from the field of information retrieval for log abstraction in information security. To this end, we evaluate three feature extraction techniques and three clustering algorithms on four different security datasets for anomaly detection. Results demonstrate that these techniques have a role to play for log abstraction in the form of extracting syntactic features which improves the identification of anomalous minority classes, specifically in homogeneous security datasets.

ACM CCS: Security and Privacy → Intrusion/anomaly detection

Keywords: information security, log abstraction, syntactic features, clustering

1 Introduction

In most operational networks, all messages and alarms from distributed network/service elements are logged with timestamps into data logs. The logs from different systems can then be pooled together in a central database for subsequent analysis. While log analysis has been studied in the literature in multiple contexts, we maintain that previous approaches have adopted techniques that fail to address the broad scope of log files and/or are not specific enough for addressing security issues.

In particular, benchmarking and replication studies performed in the literature demonstrate that their generalizability in terms of heuristics, accuracy, scalability and system/data independece in homogeneous and heterogeneous environments face several challenges [1, 2, 3]. Thus, in this research we explore the use of lightweight syntactical feature extraction techniques as an alternative approach for log abstraction to detect anomalies in security log files.

Techniques for syntactical feature extraction (such as ratios of function words etc.) are well known and well studied in terms of scalability and data independence in the information retrieval field [4]. Thus, our aim is to understand their utility in the security field. The underlying objective is to identify the form of message tokens for log data abstraction. To this end, we benchmark three different (unsupervised) clustering algorithms using three syntactic feature extraction techniques. Our approach is evaluated over four datasets representing both homogeneous and heterogeneous environments.

To achieve this, we expand on Gallagher et. al's work [4] to study: (i) Simple character/word counts; (ii) Basic Term Frequency Inverse Document Frequency (TFIDF); and (iii) TFIDF + Synthetic Minority Oversampling Technique (SMOTE) as syntactical feature extraction techniques. Moreover, we evaluate these techniques using clustering (unsupervised learning) algorithms, namely K-Means, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Expectation Maximization (EM), over four datasets – ECML/PKDD 2007, ISOT-CID Day 1 and Day 2, and Web Apache Access log files. To the best of our knowledge, this is the first time such a study is undertaken using these feature extraction techniques as an anomaly detection (clustering) approach over security datasets.

The rest of the paper is organized as follows. Section 2 summarizes the related works in the literature. Section 3 details the methodology followed, including the feature extraction techniques, clustering algorithms, and datasets employed. Evaluation and results are presented in Section 4. Finally, conclusions are drawn and future work is discussed in Section 5.

2 Related Works

The collection of log messages to characterize the operation of deployed services and applications is an integral component of forensic analysis for the identification and understanding of security incidents. Many approaches have been proposed in the literature for automatic

log parsing and abstraction by analyzing different network, system and service related data. To this end, data sources could be categorized into three types: (i) application/service log files [5, 6], (ii) operating system log files [7, 8], and (iii) network traffic log files [9, 10]. Among these, operating system log files have received the most attention for automatic log parsing and abstraction algorithms in order to analyze root causes of incidents, detect alerts, and monitor system performance.

Recently, Zhu et. al. [1] evaluated 13 algorithms that were introduced between 2003 and 2018. The algorithms were compared according to the execution time, the number of required pre-processing steps, online/offline, scalability to the log size, capability to parse under different conditions, and their open source availability. Also, they were evaluated and ranked based on their performance using a set of annotated datasets, including application/service and operating system log files. Moreover, Copstein et. al. replicated the analysis of those algorithms on the same log files and security log files [3]. The assessment focused on the capacity for parsing and abstraction of log files from the perspective of information security and forensic analysis. They confirm that the replication experiments obtain similar performance in most cases as demonstrated by Zhu et. al. [1]. However, some of the obtained results were significantly different than those in the original paper. This might be indicative of the lack of robustness in some of these algorithms [11].

El Masri et. al. [2] followed a similar approach, this time considering 17 log analysis algorithms. Of those, 12 were the same as the ones used in Zhu et. al. and Copstein et. al. works [1, 3]. However, the remaining five algorithms included techniques from machine learning and natural language processing. Moreover, El Masri et. al. did not run the 17 algorithms on datasets to measure their accuracy. Instead, they compared the algorithms using a set of specific criteria, namely scalability, system independence, requirement of data pre-processing, and the time complexity of the algorithm (computational cost). Their results indicate that while machine learning and natural language processing-based techniques employed have challenges in the scalability criteria, they seem to demonstrate opportunities in terms of system as well as delimeter indepedence.

Bhamare et. al. employed supervised learning algorithms to automatically parse and classify network traffic log files [12]. They benchmarked Decision Tree, Support Vector Machine, and Naive Bayes classifiers on UNSW and ISOT cloud intrusion detection datasets using packet based features. Their results show that while the classification algortihms could reach over 90% identification of normal and attack packets on UNSW log files, they can at most reach 54% true positive in attack detection on ISOT log files.

Andriamanalimanana et. al. also employed ISOT cloud intrusion detection dataset for evaluating the Kullback-Leibler divergence (of probability distributions) for designing anomaly scores [13]. However, they only use part of the ISOT dataset, namely the output of *iostat*, *vmstat -d* and *vmstat -a* commands to detect anomalies. Several anomalies were reported, however, the exact accuracy of the proposed method was not measured. In a second synthetic dataset they were able to measure the accuracy in terms of recall of anomalies which reached up to 90%.

Nguyen et. al. design an adaptive intrusion detection system with 10% higher accuracy than the best of four different baseline classifiers, namely Naive Bayes, Bayes Network, Decision Stump and RBF Network [14]. They combined the outputs of the classification systems by using the online learning framework. They evaluated their approach over benchmarking web application security using ECML/PKDD HTTP dataset and CSIC HTTP, a dataset they generated. Their results showed an accuracy of around 91% and 93% on CSIC and ECML/PKDD datasets, respectively. However, it should be noted here that during the ECML/PKDD 2007 HTTP attack detection challenge, the highest performances reported for classification systems were 83%, 78% and 80% for precision, recall and F-Score, respectively [15].

In summary, while log abstraction algorithms seem to work accurately on operating system and application log files as shown by Zhu et. al, El Masri et. al. and Copstein et. al., they do not work accurately for information security log files [3]. On the other hand, the literature on log abstraction for information security increasingly employs handcrafted features dependent on delimeters and log characteristics with supervised classification systems. These result in research gaps in terms of generalizability and scalability to new systems and behaviours as the services and networks we use grow. Thus, in this work, we aim to explore lightweight syntactic features using an anomaly detection based approach in order to study these gaps.

3 Methodology

Log Abstraction algorithms will often be observed to be sensitive to the number of tokens (words) present in log lines. In other words, most of these algorithms will group lines of different sizes into different sets, despite any other perceived similarity. This demonstrates a core idea behind log abstraction algorithms: the value of *form* over *function*. In other words, it seems that for the majority of log abstraction algorithms the *syntax* is more important than the *semantics*.

For log files that are composed of syntactically different strings – different wording, order of tokens – syntactic analysis is, arguably, the most efficient form for summarizing the logs by similarity. With this in mind, we explore the efficacy of attributes specifically related to syntax,

and how they perform from a clustering standpoint. As the next step, we experiment with these features using techniques from information retrieval such as vector representation and lightweight natural language processing. We maintain that combining lightweight information retrieval techniques with clustering has the potential to enable a more scalable and system/delimeter-independent approach to anomaly detection on information security log files. This also allows us to perform experiments over a controlled set of features, as opposed to making the decision of feature selection solely based on the output of the log abstraction algorithm.

In [4], Gallagher et. al. studied the use of a vector space model, a commonly assumed approach for information retrieval, on HTTP attack classification. Their results showed that it was possible to achieve a high performance in the classification of information security related data, namely the ECML/PKDD 2007 dataset [16]. To this end, they used syntactic feature extraction techniques such as TFIDF [17]. However, they achieved this by using the labels of the dataset to create *a priori* document models for each label (category) present in the dataset, i.e. each attack category as well as the normal category of data represented a document model. Their classification approach, while achieving a high performance, also comes with a disadvantage: It could only define 'terms' for a 'category' present in the training data, thus, log files with new terms and categories would not be represented completely.

In this work, we instead adopt feature engineering approaches to identify a set of syntactical attributes for anomaly detection. To this end, we employ clustering techniques as opposed to classification, albeit by computing a *single* document model, as opposed to one document model per class (label). We aim to explore whether similar performance could be achieved under these more relaxed conditions. This is relevant given that by separating document models per class, one is introducing a form of *a priori* class segmentation, which reduces the requirement of any trained model to discover this step on its own. Given that we employ an anomaly detection approach via clustering, we do not use labels for feature selection or training purposes. Thus, we experiment with several syntactical attribute extraction techniques, such as TFIDF, and measure their performance properties under a clustering approach over three security related datasets:

- **The ECML/PKDD 2007 Discovery Challenge Dataset** [16] was created and published to be used as training/testing data for classifiers for web attack detection. This publicly available dataset contains approximately 50,000 entries that are distributed across eight classes as shown in Table 1.
- **The ISOT-CID Dataset** [18] was created and published to be used as training/testing data for classifiers on cloud environments for intrusion detection purposes. This dataset contains 10 days of labelled network traffic at VM and hypervisor levels, system logs, performance data (e.g. CPU utilization), and system calls. The first and second days of the first collection period were selected for our evaluations. Both were randomly sampled to represent a balanced distribution of malicious to benign records: 3,170 entries for day 1, and 6,293,326 entries for day 2, distributed across two classes as shown in Table 2.
- **The Web Apache Access Logs** [19] was created and published to provide training/testing data for web attack detection. This set is an annotated Apache access log file with 37,693 entries, distributed across three – originally named in Indonesian – classes: Safe (Aman), Danger (Bahaya), and Suspected (Dicurigai) as shown in Table 3.

In this context, both the ECML/PKDD 2007 and the Web Apache Access Logs datasets are *homogeneous*, that is, they contain entries (log lines) from a single deployed application, in this case, Apache web servers. The ISOT-CID dataset, on the other hand, is *heterogeneous*, meaning that it contains entries from multiple applications deployed in a common environment, in this case, a cloud infrastructure.

Table 1: Summary for the ECML/PKDD 2007 dataset

Class	#	%
C1 - Normal	35,006	69.84%
C2 - Cross-Site Scripting	1,825	3.64%
C3 - SQL Injection	2,274	4.53%
C4 - LDAP Injection	2,279	4.54%
C5 - XPATH Injection	2,279	4.54%
C6 - Path Traversal	2,295	4.57%
C7 - Command Execution	2,302	4.59%
C8 - SSI Attacks	1,856	3.70%
Total:	50,116	100%

Table 2: Summary for the ISOT-CID dataset

	Day 1		Day 2	
Class	#	%	#	%
Benign	1,585	50%	3,112,457	49%
Malicious	1,585	50%	3,180,869	51%
Total:	3,170	100%	6,293,326	100%

Table 3: Summary for the Web Apache Access Logs dataset

Class	#	%
AMA - Safe	29,778	79%
BAH - Danger	4,965	13.17%
DIC - Suspected	2,950	7.83%
Total:	37,693	100%

3.1 Syntactical Feature Extraction Techniques Employed

As discussed earlier, in this work we evaluate three syntactical feature extraction techniques with minimum a priori information on our datasets. These are summarized as the following:

3.1.1 Syntactical Feature Extraction Technique 1: Baseline

In order to establish a baseline performance for clustering using syntactic features, we design the first approach with four syntactic features from each entry (log line) in the datasets. These features are chosen to be the baseline due to their independence of any external model or pre-processing in order to be calculated:

- **Alphanumeric Ratio**: The ratio of alphanumeric characters over the total number of characters in the log line;
- **Average Character**: The average of the numeric value of all characters in the log line;
- **Character Count**: The number of characters in the log line;
- **Word Count**: The number of words in the log line (as separated by a whitespace);

It should be noted here that these techniques are the most basic forms of extracting syntactic features. While they may not be optimal in some scenarios – URL decomposition, for example, due to the lack of whitespaces and the presence of character separators – they should be able to produce meaningful values in most of them.

3.1.2 Syntactical Feature Extraction Technique 2: TFIDF

Term Frequency Inverse Document Frequency (TFIDF) [17] is a technique used in the field of natural language processing to generate representations of text in the form of vectors of values. The technique is based on the existence of a document model which contains one or more reference documents.

Given a document, d, and a document model, D, composed of N documents; a TFIDF vector is calculated by calculating $tfidf(t, d, D)$ for every term, t, as follows:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

$$tf(t, d) = log(1 + freq(t, d))$$

$$idf(t, D) = log(\frac{N}{count(d \in D : t \in d)})$$

In other words, $tfidf(t, d, D)$ increases as the number of times that t appears in either the given document or in the document model decreases – and vice-versa. That is to say that a term with a high TFIDF value is an infrequent term.

Given the satisfactory results reported in [4], we chose to use TFIDF as part of the second approach to feature extraction. To this end, we start by building a document model composed of all the entries in the dataset – where one entry is seen as one document. Next, for each entry in the dataset, we calculate the TFIDF values for its terms and extract the **minimum value**, the **maximum value**, and the **average value**. Unlike [4], we do not have a separate document model for each class in the dataset. This not only enables us to minimize the requirement for *a priori* information such as labels – which are rarely available in practice for information security – but also enables our approach to be used for any log data since no handcrafted features are necessary. Thus, we believe that by doing so we are more accurately measuring the model's capacity to find discrepancies between the existing entries and their behaviours as they were encountered in the real world.

3.1.3 Syntactical Feature Extraction Technique 3: TFIDF + SMOTE

It is apparent that class imbalance in datasets can introduce bias towards the majority class [20]. To address this challenge, for the third feature extraction approach, we employed the Synthetic Minority Oversampling Technique (SMOTE) together with TFIDF. SMOTE [21], is a technique that uses known samples of minority classes to synthetically produce new exemplars for that given class. The production of such instances of the class maintains its overall characteristics, which reduces the probability of introducing outliers. Employing this approach means that we 'flex' our self imposed *minimum a priori information* constraint for preprocessing purposes.

By making use of SMOTE, we rebalance a given dataset by producing new samples for the non-majority classes. For example, in the case of ECML/PKDD 2007 dataset, minority classes C2 to C8 were balanced by oversampling such that they end up with 10,000 instances each. Additionally, for the majority class, C1, we randomly resampled the available instances to select 10,000 exemplars as well. As for the Web Apache Access dataset, we introduce new samples for the non-majority classes – BAH (danger) and DIC (suspicious) – such that they ended up with 11,000 instances each. The majority class – AMA (safe) – was randomly resampled to select 11,000 exemplars as well.

3.2 Clustering Algorithms Employed

In this research, we evaluate three clustering – unsupervised learning – algorithms for anomaly detection:

- **Simple K-Means** is computationally efficient and easy to implement. As such, it is one of the most popular iterative clustering algorithms [22]. The algorithm begins by randomly initializing k cluster centroids. Data points are then assigned to the cluster

with the smallest Euclidian distance between the data point and the cluster centroid. Centroids are then recomputed according to the mean position of all points in a given cluster. The algorithm converges when the centroids stabilizes or the movement is less than the given movement threshold. In this context, the most important parameter to choose is the number of clusters, k. The other parameters to be chosen are the number of iterations and the maximum iterations, which affect the stopping criteria of the algorithm.

- **DBSCAN** is a clustering algorithm that operates on the premise that clusters are groups of densely packed data points separated by less dense space [23]. It considers a specified number of nearby points (`MinPts` or `min_samples`) as a cluster core if they fall within a specified distance (ε, or `eps`). Clusters are expanded to border points accessible from at least one of the core points. Points that are at least ε away from a core point are not added to the cluster, and instead marked as noise. These values must be carefully chosen so as not to create one large cluster, few, or no clusters. One issue in applying DBSCAN to data is that the density of data points is not necessarily consistent across a dataset, and the choice of ε can prioritize more dense groupings over less dense, and vice-versa.
- **EM** is an approach by which one can iteratively find the parameters of statistical models that approximate the data to be clustered [24]. The data is assumed to be modelled by Gaussian distributions, where the number of distributions is equal to the number of clusters. Starting from initial models, the algorithm alternates between expectation and maximization steps. In the expectation step, data points are assigned to a model based on a probabilistic calculation. Then, the maximization step computes parameters for the models to maximize the log-likelihood of the data based on the assignments of the previous step. Thus, the iterative model is used to find (local) maximum likelihood or maximum a posteriori (MAP) estimates of parameters. The algorithm is considered to have converged when the updated models do not substantially change between iterations.

These were chosen due to the different techniques they employ for clustering, that is the smallest Euclidean distance on K-Means, point density on DBSCAN, and MAP estimates for EM.

4 Evaluations and Results

In order to evaluate each one of the syntactical feature extraction techniques, we trained a model using the aforementioned algorithms to cluster the data on each dataset. Our goal is to understand how well clusters match the accompanying labels (groundtruth) post training. In order to achieve this, we make use of Waikato Environment for Knowledge Analysis (WEKA) [25] – a tool for the execution of knowledge analysis algorithms developed by the University of Waikato – to run three different clustering (unsupervised learning) algorithms to analyze the datasets. It should be noted here that, for all clustering algorithms, we used the default set of parameters recommended by WEKA. The only exception being the value of K for the K-Means algorithm, set to N, to match the number of classes in the dataset being evaluated (if known). We further increase the value of K to see the effects of increasing the number of clusters. In scenarios where the number of classes is unknown, this value could be determined empirically.

Post training, to measure the performance, each of the clusters is associated with one of the original dataset labels such that a cluster corresponds to the label with highest cluster association. In other words, for each cluster, we count the number of its members (log lines) that are associated with each possible label in the dataset (e.g. how many members belong to label C1), and associate the overall cluster with the label whose number of members is the highest. This allows one to gather, for each cluster: (i) the number of instances (i.e. log lines) where the instance label matches the associated cluster's label (true positives – TP); (ii) the number of instances where the instance label does not match the associated cluster's label (false positives – FP); (iii) the number of instances where the instance label matches the current cluster's label, but were not included in it (false negatives – FN); and (iv) the number of instances where the instance label does not match the current cluster's label, and were not included in it (true negatives – TN). After training, this enables us to calculate the performance of each clustering algorithm with each feature extraction technique in the form of precision (P), recall (R), and F1-score (F) for all of the available labels (categories).

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN}$$

$$F = \frac{2 \times P \times R}{P + R}$$

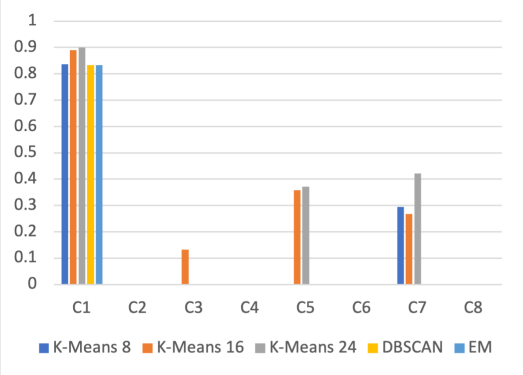
Note that the process of associating a cluster with a label is only performed for the sake of performance evaluation. In a production scenario, the dataset labels would not be used during training, as previously mentioned, because most of the time they are not known.

4.1 Experiment 1

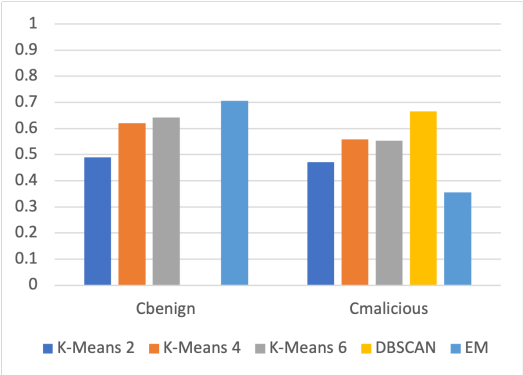
The resulting F-Score for the experiments using the baseline syntactical feature extraction technique 1 can be seen for each dataset in Figure 1.

For the ECML/PKDD 2007 dataset and the Apache Access dataset – given that anomalous entries contain more

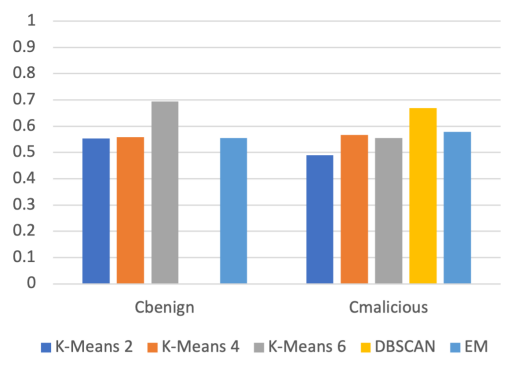
Figure 1: F-Score values for experiment 1 using the baseline technique for each dataset



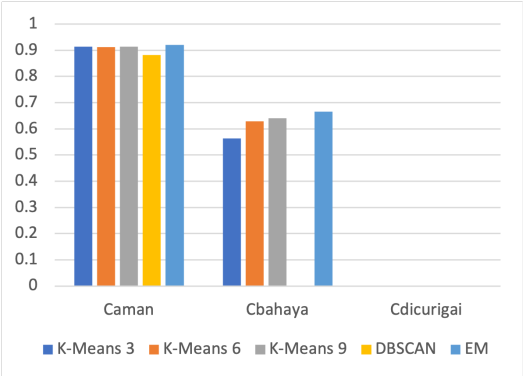
(a) ECML/PKDD'07 Dataset



(b) ISOT-CID Day 1 Dataset



(c) ISOT-CID Day 2 Dataset



(d) Apache Access Dataset

than one class – we also report the results where all attacks are in a single group, i.e. anomolous, as seen in Figures 2 and 3.

In this first set of experiments, we note that, for the K-Means algorithm, the performance showed a tendency towards improvement with increasing values of k. The EM algorithm, in the case of the ISOT-CID Day 2 and the Web Apache Access Logs datasets, performed similarly to K-Means, whereas for ISOT-CID Day 1 it showed a less balanced class distribution. For the ECML/PKDD 2007 dataset, EM returned clusters representative of only 1 class. DBSCAN also returned this result under all datasets.

When looking at the results as only normal and anomalous classes, we can more clearly see the improvement with increases in the value of k for the the ECML/PKDD 2007 dataset. Given that both DBSCAN and EM could only differentiate one class in this experiment, they were not analyzed from this perspective.

The Apache Access dataset, in turn, did not benefit as much from the increase in the value of k for the K-means algorithm. The results with the most balanced class distribution comes from the EM algorithm, despite not being able to discern the third class – Suspicious (Dicurigai). Once again, DBSCAN was not analyzed further because of the poor performance on this dataset.

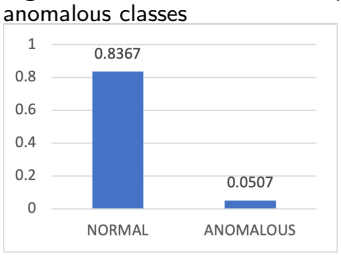
4.2 Experiment 2

A second set of experiments is performed using the second syntactical feature extraction technique, i.e. including the TFIDF features. The resulting F-Score for these experiments can be seen for each dataset in Figure 4. The results for the ECML/PKDD 2007 dataset and the Apache Access dataset considering only normal and anomalous classes can be seen in Figures 5 and 6.

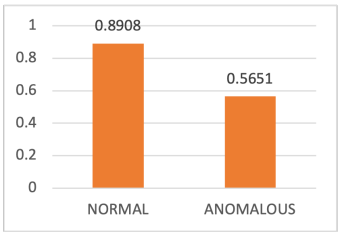
Under this second set of experiments, we can see the benefit of the TFIDF features in achieving a better class distribution. Both K-Means and EM showed improvements in this respect while maintaining similar performances when compared to the previous set of experiments. DBSCAN, however, maintained its previous class distribution and performance.

A noteworthy result here is the significant improvement seen in the results for the ISOT-CID Day 1 dataset. With the exception of DBSCAN, all algorithms had similar or improved performance when compared to the previous set of experiments. Moreover, looking at the results as normal versus anomalous classes, we see that they are comparable to the previous set of experiments. This is because any improvement in the balance of class distribution gets abstracted by the more general anomalous class.

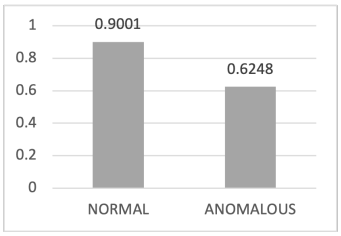
Figure 2: F-Score values for experiment 1 using the baseline technique over ECML/PKDD dataset considering only normal and anomalous classes



(a) KMeans 8

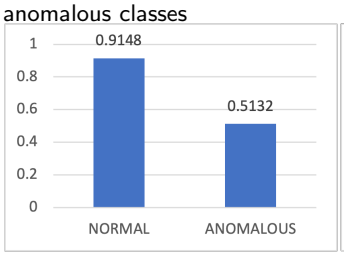


(b) Kmeans 16

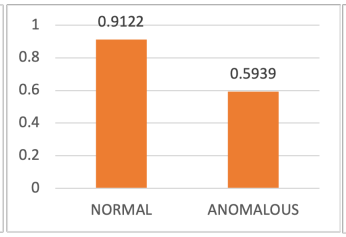


(c) Kmeans 24

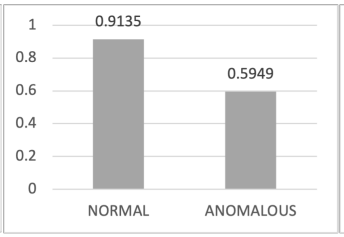
Figure 3: F-Score values for experiment 1 using the baseline technique over Apache Access dataset considering only normal and anomalous classes



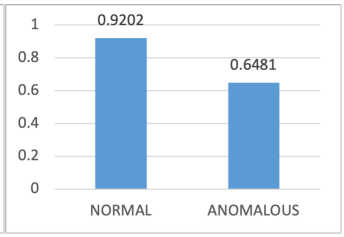
(a) KMeans 3



(b) Kmeans 6



(c) Kmeans 9



(d) EM

4.3 Experiment 3

The final set of evaluations used syntactical feature extraction technique 3 (SMOTE over sampling) on two of the four datasets. That is to say, since both days of the ISOT-CID dataset are balanced, there is no need for oversampling. These results can be seen in Figure 7. The results for normal versus anomalous classes can be seen in Figures 8 and 9.

Here, we see a major improvement for the K-Means algorithm. It was able to reach similar class distribution and performance as the previous experiment, albeit with a smaller number of clusters necessary (smaller value of k). EM showed improvement in class distribution for the ECML/PKDD 2007 dataset and similar performance for the Web Apache Access Logs dataset. DBSCAN yielded similar performance as the previous two sets of experiments.

4.4 Analysis

In the evaluations above, we can see that the K-Means algorithm tends to perform better as the value of k increases. This results in performance increases especially in the minority classes, which enables the algorithm to more affectively recognize anomalous classes.

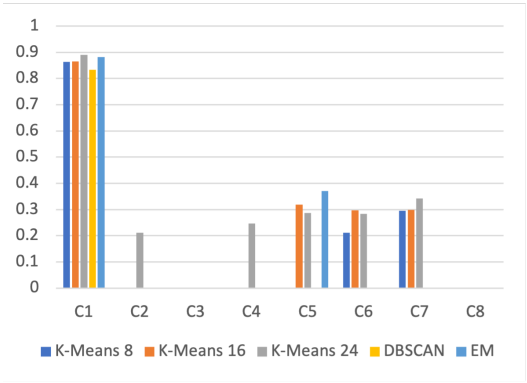
In the case of the ECML/PKDD 2007 dataset, the presence of the TFIDF features was crucial in the process of forming clusters that better represent the minority classes, whereas the use of SMOTE helped further in the representation of class distributions. For the Web Apache Access Logs dataset, use of TFIDF+SMOTE demonstrates a turning point where the algorithm was able to find clusters for all three classes for the first time. The ISOT-CID Day 1 dataset showed significant improvement with the addition of the TFIDF features, whereas Day 2, despite showing some improvement, did not benefit much from it.

The DBSCAN algorithm, throughout the experiments, was not able to produce clusters representative of more than one class at a time. Being it a density-based clustering algorithm, we observe that most instances did not seem to be placed in dense clusters in the N-dimensional space created by the extracted syntactic features.

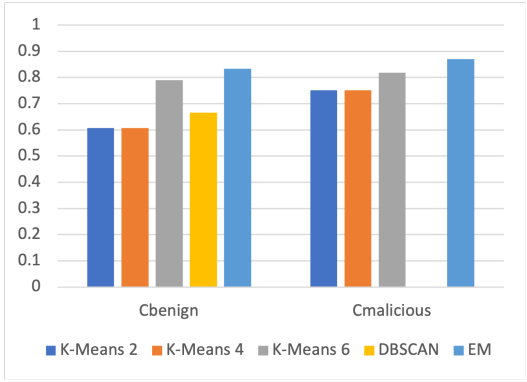
Lastly, the EM algorithm showed a similar trend as the K-Means algorithm in the baseline technique for the majority class. However, it was not able to identify minority classes in the ECML/PKDD 2007 dataset and the Web Apache Access Logs dataset. For the ISOT-CID Day 1 dataset, there was a decrease in performance when identifying the malicious class, whereas in the ISOT-CID Day 2 this issue was not evident and the performance was similar to that of the K-Means algorithm. The presence of the TFIDF features provided improvements to the results of the EM algorithm in terms of identifying the minority classes both for the ECML/PKDD and Web Apache Access Logs datasets, as well as significant improvements in identifying both classes in the ISOT-CID Day 1 dataset.

Among the feature extraction techniques employed, it is observed that TFIDF+SMOTE provides the best evaluations on homogeneous datasets while basic TFIDF enables the best detection of minority classes on all datasets. Meanwhile, the K-Means algorithm consistently achieved better performance with higher K values. Itvis unclear whether the performance would continue improving if we kept on increasing the value of K. Moreover, we believe that there is a balance between the number of features available and the number of clusters requested,

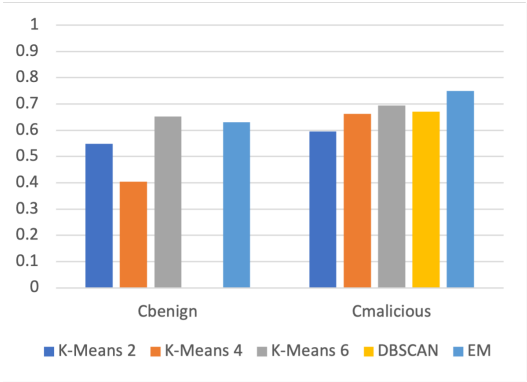
Figure 4: F-Score values for experiment 2 using the TFIDF technique over datasets



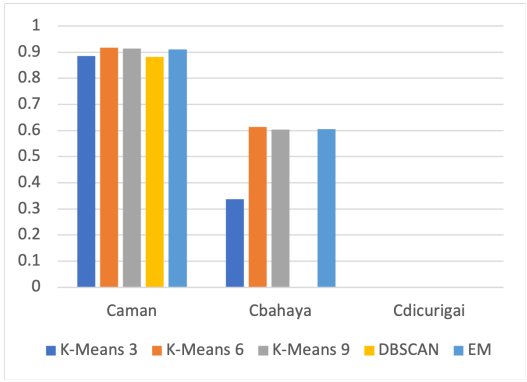
(a) ECML/PKDD 2007 Dataset



(b) ISOT-CID Day 1 Dataset



(c) ISOT-CID Day 2 Dataset



(d) Apache Access Dataset

i.e. post training response time will be a function of the number of features and clusters employed.

When compared to related works in the literature, we can observe that the baseline approach showed similar performance to [12] by reaching around 54% F-score in attack detection on the ISOT-CID dataset (both days) using the K-means algorithm. Conversely, in experiment 2, the F-score gets as high as 80% for attack detection on the ISOT-CID Day 1. As for the ECML/PKDD 2007 dataset, while the highest F-score reported in [15] was 80%, our method was able to reach 85% F-score for detection of normal entries, and 66% F-score for detection of attacks using approach 3 with the K-means algorithm.

5 Conclusion

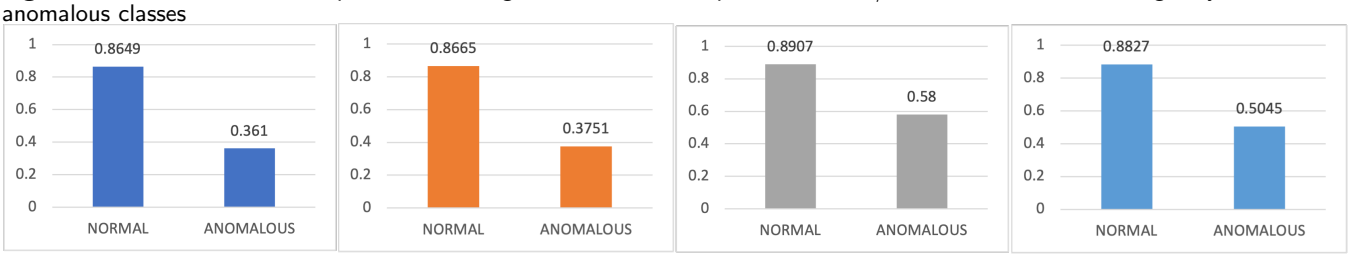
The efficient analysis of log messages has become increasingly important with the ever-growing scale of cyber attacks and equally growing collection of tracked information for forensic analysis. To this end, techniques based on ML have grown in popularity due to their promise of meeting the performance required by such systems. However, most of these techniques still rely on the selection of suitable features to be extracted from the log messages in order to reach their full potential.

Given the observed success of syntactic techniques such as TFIDF in classifying network security data [17], in this work we explore the impact of using syntactic features for anomaly detection and their utility for information security. In doing so, our goal is to understand the effect of the *form* of message tokens for representing log events as features. To this end, we evaluated three feature sets based on lightweight syntactic feature extraction techniques (baseline, basic TFIDF, and TFIDF with SMOTE) on four security datasets (two homogeneous and two heterogeneous) using three clustering algorithms, namely K-Means, DBSCAN and EM.

Our results indicate that the representation of log files using basic TFIDF enables the detection of anomous behaviours (minority classes in ECML/PKDD and Web Apache Access datasets) more than the baseline technique on the homogeneous datasets. Augmenting TFIDF with SMOTE indicates even better performance on anomoulous (minority) classes by balancing the training data across all clustering techniques. Moreover, the K-Means algorithm performs better or is competetive to DBSCAN and EM algorithms, irrespective of the feature set or dataset used. On the other hand, the effect of basic TFIDF versus the baseline technique on the heterogonous and balanced datasets seems to be less consistent.

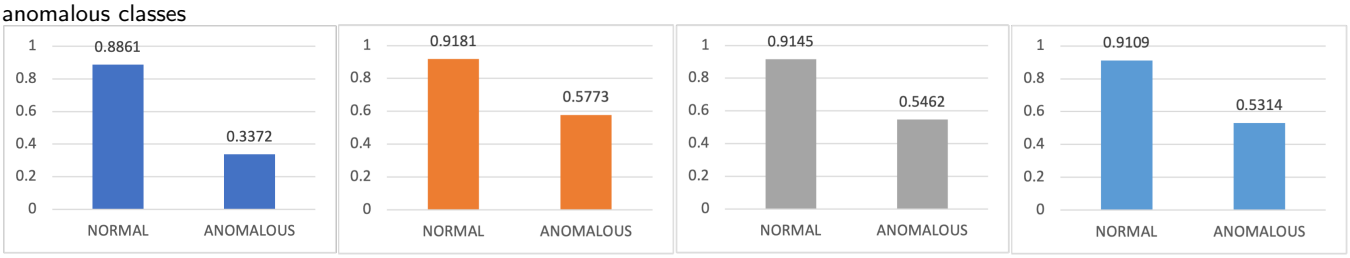
In some scenarios, we observe that challenges exist in differentiating one type of attack from another – such as attack classes C2 and C8 for the ECML/PKDD 2007 dataset. We believe that these cases represent vulnerabilities that are less reliant on syntax attributes than the

Figure 5: F-Score values for experiment 2 using the TFIDF technique over ECML/PKDD dataset considering only normal and anomalous classes



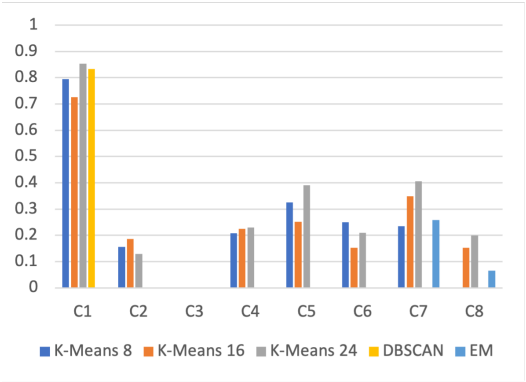
(a) KMeans 8 (b) Kmeans 16 (c) Kmeans 24 (d) EM

Figure 6: F-Score values for experiment 2 using the TFIDF technique over Apache Access dataset considering only normal and anomalous classes

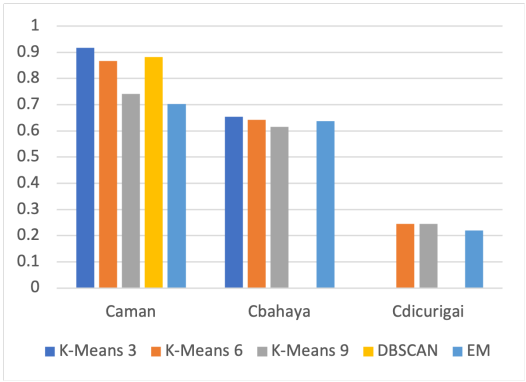


(a) KMeans 3 (b) Kmeans 6 (c) Kmeans 9 (d) EM

Figure 7: F-Score values for experiment 3 using the TFIDF+SMOTE technique over datasets



(a) ECML/PKDD 2007 Dataset



(b) Apache Access Dataset

other, more easily identified ones. For example, a path traversal attack can be perceived as more impactful in terms of syntax – given the presence of multiple path separators (/") and/or relative path indicators (".") – than a command execution attack – where most commands are indistinguishable from regular words. This could be a limitation of using syntax-related features with classifiers, where the classification is highly dependent on how much the vulnerabilities being searched for impact the syntax of the logs. Further analysis is necessary to understand the reasons behind these phenomena. When compared to similar work in the literature, we observe comparable performance yielded by our method despite only using lightweight syntactical features.

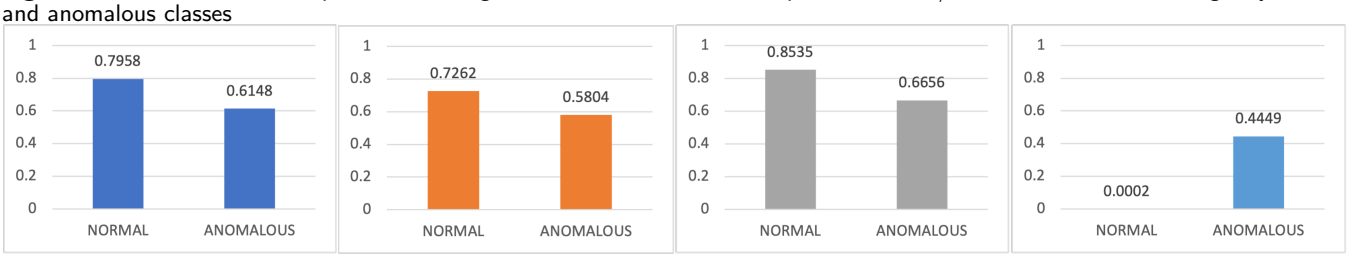
Overall, these results indicate that log abstraction using lightweight syntactic feature extraction techniques on balanced training datasets improves the performance of clustering algorithms, specifically K-Means on homogeneous security datasets.

Future research will explore augmenting the syntactical features with semantic features by incorporating the security heuristics proposed in [3]. We believe this could provide benefits in terms of explainability and root-cause analysis for anomaly detection in information security. Moreover, we will explore the use of different syntactical features on heterogeneous datasets to better understand their effects. Further study into homogeneous and heterogeneous environments is desirable. To this end, future research in heterogeneous cloud environments would be necessary given the ever growing enterprise application deployment on these platforms.

Acknowledgement

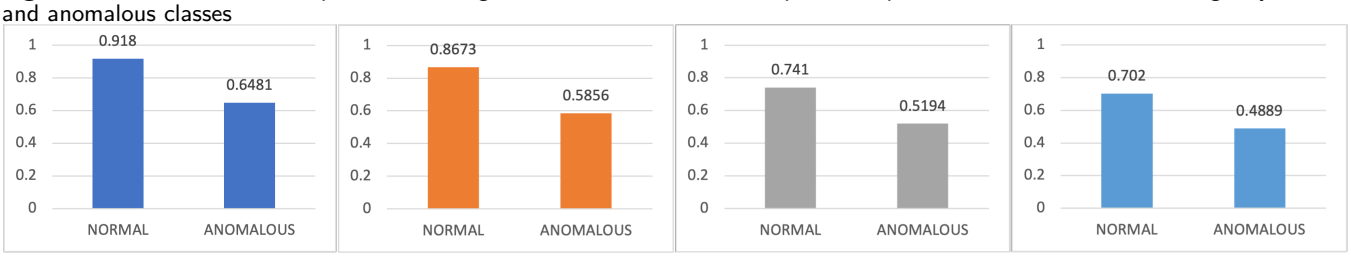
This research was enabled by the support of the Natural Science and Engineering Research Council of Canada Alliance Grant and 2Keys Corporation. The first author gratefully acknowledges the support by the province of Nova Scotia.

Figure 8: F-Score values for experiment 3 using the TFIDF+SMOTE technique over ECML/PKDD dataset considering only normal and anomalous classes



(a) KMeans 8 (b) Kmeans 16 (c) Kmeans 24 (d) EM

Figure 9: F-Score values for experiment 3 using the TFIDF+SMOTE technique over Apache Access dataset considering only normal and anomalous classes



(a) KMeans 3 (b) Kmeans 6 (c) Kmeans 9 (d) EM

The research is conducted as part of the Dalhousie NIMS Lab at: https://projects.cs.dal.ca/projectx/.

Literature

[1] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "Tools and benchmarks for automated log parsing," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2019, pp. 121–130.

[2] D. El-Masri, F. Petrillo, Y.-G. Guéhéneuc, A. Hamou-Lhadj, and A. Bouziane, "A systematic literature review on automated log abstraction techniques," *Information and Software Technology*, vol. 122, p. 106276, 2020.

[3] R. Copstein, J. Schwartzentruber, N. Zincir-Heywood, and M. Heywood, "Log abstraction for information security: Heuristics and reproducibility," in *The 16th International Conference on Availability, Reliability and Security*, ser. ARES 2021. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3465481.3470083

[4] B. Gallagher and T. Eliassi-Rad, "Classification of http attacks: a study on the ecml/pkdd 2007 discovery challenge," Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), Tech. Rep., 2009.

[5] H. Dev and Z. Liu, "Identifying frequent user tasks from application logs," in *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, ser. IUI '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 263–273. [Online]. Available: https://doi.org/10.1145/3025171.3025184

[6] K. Savitha and M. Vijaya, "Mining of web server logs in a distributed cluster using big data technologies," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 5, no. 1, 2014.

[7] C. Lonvick, "Rfc3164: The bsd syslog protocol," 2001.

[8] A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "A lightweight algorithm for message type extraction in system application logs," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 11, pp. 1921–1936, 2012. [Online]. Available: https://doi.org/10.1109/TKDE.2011.138

[9] F. Haddadi and A. N. Zincir-Heywood, "Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification," *IEEE Syst. J.*, vol. 10, no. 4, pp. 1390–1401, 2016. [Online]. Available: https://doi.org/10.1109/JSYST.2014.2364743

[10] R. Alshammari and A. N. Zincir-Heywood, "The impact of evasion on the generalization of machine learning algorithms to classify voip traffic," in *21st International Conference on Computer Communications and Networks, ICCCN 2012, Munich, Germany, July 30 - August 2, 2012*. IEEE, 2012, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ICCCN.2012.6289243

[11] D. C. Le and N. Zincir-Heywood, "A frontier: Dependable, reliable and secure machine learning for network/system management," *J. Netw. Syst. Manag.*, vol. 28, no. 4, pp. 827–849, 2020. [Online]. Available: https://doi.org/10.1007/s10922-020-09512-5

[12] D. Bhamare, T. Salman, M. Samaka, A. Erbad, and R. Jain, "Feasibility of supervised machine learning for cloud security," *CoRR*, vol. abs/1810.09878, 2018. [Online]. Available: http://arxiv.org/abs/1810.09878

[13] B. Andriamanalimanana, A. Tekeoglu, K. Bekiroglu, S. Sengupta, C. Chiang, M. Reale, and J. E. Novillo, "Symmetric kullback-leibler divergence of softmaxed distributions for anomaly scores," in *Conference on Communications and Network Security*. IEEE, 2019, pp. 1–6.

[14] H. T. Nguyen and K. Franke, "Adaptive intrusion detection system via online machine learning," in *International Conference on Hybrid Intelligent Systems*. IEEE, 2012, pp. 271–277.

[15] C. Raissi, J. Brissaud, G. Dray, P. Poncelet, M. Roche, and M. Teisseire, "Web analyzing traffic challenge: description and results," in *Proceedings of the ECML/PKDD*, 2007, pp. 47–52.

[16] ECML/PKDD, "Ecml/pkdd 2007 discovery challenge," September 2021, https://gitlab.fing.edu.uy/gsi/web-application-attacks-datasets/-/tree/master/ecml_pkdd.

[17] A. Aizawa, "An information-theoretic perspective of tf–idf measures," *Information Processing & Management*, vol. 39, no. 1, pp. 45–65, 2003.

[18] University of Victoria, "Isot-cid cloud security," October 2021, https://www.uvic.ca/ecs/ece/isot/datasets/

cloud-security/index.php?utm_medium=redirect&utm_source=/engineering/ece/isot/datasets/cloud-security/index.php&utm_campaign=redirect-usage.

[19]

[20] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[22] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[23] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996, p. 226–231.

[24] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

Rafael Copstein graduated Bachelor of Computer Science at the Pontifical Catholic University of Rio Grande do Sul (PUCRS) - Brazil - in 2019, as first of his class as recognized by the Brazilian Computer Society (SBC). In that same year, he was awarded a scholarship at Dalhousie University - Canada - to pursue a Master's Degree in Computer Science under the supervision of Dr. Nur Zincir-Heywood in the area of Computer Networks. In 2021, as an invitation from his supervisor, he upgraded his degree into a Ph.D. in Computer Science. Shortly after, he was awarded the Nova Scotia Graduate Scholarship (NSGS) due to the relevance of his research to the province of Nova Scotia and due to his academic achievements.

Address: Dalhousie University, Faculty of Computer Science, 6299 South Street, Halifax, NS - Canada
E-Mail: rafael.copstein@dal.ca

Egil Karlsen is a Masters of Computer Science student at Dalhousie University. He received his Bachelor of Computer Science with First Class Honors from Dalhousie University where he performed research in the area of vulnerability analysis, authentication and authorization services specifically in the OAuth protocol. Research interests include applications of machine learning in cyber security.

Address: Dalhousie University, Faculty of Computer Science, 6299 South Street, Halifax, NS - Canada
E-Mail: egil.karlsen@dal.ca

Dr. Jeff Schwartzentruber holds the position of Principal Data Scientist and Research Lead at 2Keys Corporation. Dr. Schwartzentruber received his PhD in Mechanical Engineering from Ryerson University with a focus on analytical process modelling and is a fellow of the Ontario Centre of Excellence. In his role at 2Keys, Dr. Schwartzentruber is responsible for the continued development, innovation and leadership of artificial intelligence (AI) and machine learning (ML) capabilities at the intersection of identity and access management, advanced threat analytics and response, and managed security services. Jeff's research interests include machine learning (particularly deep learning and boosted trees), real-time anomaly detection, and analytical/semi-empirical model development for security and business applications.

Address: 2Keys, 20 Eglinton Ave. W. - Suite 1500, Toronto - Ontario, Canada
E-Mail: jschwartzentruber@2keys.ca

Dr. Nur Zincir-Heywood is a University Research Professor and a Professor of Computer Science at Dalhousie University. Her research interests include machine learning for cyber security, network and service analysis. She has published over 200 fully reviewed papers and has been a recipient of multiple best paper awards. She serves as an Associate Editor of the IEEE Transactions on Network and Service Management and Wiley International Journal of Network Management. She also promotes information communication technologies to wider audiences as a tech columnist for CBC Information Morning and a Board Member on CS-Can/INFO-Can.

Address: Dalhousie University, Faculty of Computer Science, 6299 South Street, Halifax, NS - Canada
E-Mail: zincir@cs.dal.ca

Dr. Malcolm Heywood is a Professor of Computer Science at Dalhousie University, Canada. He has a particular interest in the use of machine learning to discover of simple solutions to complex tasks. His research with the tangled program graph algorithm resulted in a silver medal at the 2018 Humies Competition. Dr. Heywood is a member of the editorial board for Genetic Programming and Evolvable Machines (Springer). He was a track co-chair for the ACM GECCO GP track in 2014 and a co-chair for European Conference on Genetic Programming in 2015 and 2016 and a co-chair for the ACM GECCO GECH track in 2021 and 2022.

Address: Dalhousie University, Faculty of Computer Science, 6299 South Street, Halifax, NS - Canada
E-Mail: mheywood@cs.dal.ca