Research Article

Lightweight and Scalable Intrusion Trace Classification Using Interelement Dependency Models Suitable for Wireless Sensor Network Environment

Dae-Ki Kang

Division of Computer & Information Engineering, Dongseo University, 47 Jurye-ro, Sasang-gu, Busan 617-716, Republic of Korea

Correspondence should be addressed to Dae-Ki Kang; dkkang@dongseo.ac.kr

Received 3 May 2013; Accepted 9 June 2013

Academic Editor: S. Khan

Copyright © 2013 Dae-Ki Kang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We present a lightweight and scalable method for classifying network and program traces to detect system intrusion attempts. By employing interelement dependency models to overcome the independence violation problem inherent in the Naive Bayes learners, our method yields intrusion detectors with better accuracy. For efficient and lightweight counting of \( n \)-gram features without losing accuracy, we use a \( k \)-truncated generalized suffix tree (\( k \)-TGST) for storing \( n \)-gram features. The \( k \)-TGST storage mechanism enables us to scale up the classifiers, which cannot be easily achieved by Support-Vector-Machine- (SVM-) based methods that require implausible computing power and resources for accuracy. Experimental results on a set of practical benchmark datasets show that our method is scalable up to 20-gram with consistent accuracy comparable to SVMs.

1. Introduction

Data mining algorithms have been widely used for classifying program traces (i.e., sequences of system calls or sequences of network packets) in intrusion detection tasks. For example, in a host-based intrusion detection task, a program trace is defined as a sequence of system calls that the program invokes during its execution, whereas in a network-based intrusion detection task, a program trace can be defined as a sequence of network packets that the program transmits during its execution. As a preprocessing step for data mining algorithms on intrusion detection tasks, the \( n \)-gram (\( n \) consecutive system calls in a trace) approach [1] has been widely used for featurization of system call sequences [2–5].

However, those \( n \)-gram approaches suffer from three critical problems when applied to intrusion detection tasks.

1. **Dimensionality Issues.** Since the number of distinct system calls is usually as many as about 200, the number of distinct \( n \)-gram features increases drastically as \( n \) increases. For example, the number of distinct SunOS system calls is 183, and if a 20-gram approach is used, then the number of 20-gram features will be \( 183^{20} \), which is impractical for real world applications. The data mining algorithms that have nonlinear space complexity (such as Support Vector Machines (SVMs) [6, 7]) severely suffer from this problem.

2. **Overlap of Features.** When \( n \)-gram features are generated from an original trace using a fixed-length sliding window, one system call in the trace can be considered as many as \( n \) times in the worse case, in the resulting \( n \)-gram features [8, 9].

3. **Violation of Independence Assumption.** If the resulting intrusion detectors rely on the statistical independence assumption among features (e.g., Naive Bayes), the above-mentioned way of generating overlapped features systematically breaks the assumption.

Against these backgrounds, we applied interelement dependency models [8, 10] of \( n \)-gram features to intrusion detection tasks and compared their performance with those of widely used data mining algorithms such as Naive Bayes with \( n \)-gram features and those of SVM with \( n \)-gram features. To overcome the curse of dimensionality problem, we adapted the \( k \)-truncated generalized suffix tree storage mechanism [11, 12] to index system call traces and...
to generate counts for each $n$-gram in an efficient way. Since the features with more order information (i.e., longer $n$-gram features) from an appropriate amount of input data sets usually contribute more to classification, it is important for an intrusion detection algorithm to be scalable with the length of $n$-grams.

Experimental results on host-based and network-based intrusion detection benchmark data sets show that the proposed method outperforms the Naive Bayes learner with $n$-gram features as input, which breaks the independence assumption, on intrusion detection tasks and shows comparable accuracies and false positive rates to those of SVMs with $n$-gram features. With the suffix tree storage mechanism, we tested the performance of the classifiers up to 20-gram in the experiments, which indicates the advantage of scalability of the proposed combination of the interdependency model of $n$-gram features and suffix tree storage mechanism over the other methods. We were able to perform the experiments with $n$-grams much longer than 20-grams, but because of lack of data sets and overfitting, the results were not so significant.

The rest of the paper is organized as follows: Section 2 describes our method, Section 3 presents the experimental results, and Section 4 summarizes and concludes this paper.

2. Method

First of all, we introduce Naive Bayes with $n$-gram features (NB $n$-gram), interdependency models of $n$-Grams (IM($n$)), and SVM with $n$-gram features (SVM $n$-gram). After that, we explain suffix tree mechanism which is used to store $n$-gram features.

Before we describe each method, we formally define the intrusion detection problem as follows: Let $\Sigma = \{s_1, s_2, s_3, \ldots, s_m\}$ be a set of system calls, where $m = |\Sigma|$ is the number of system calls. Data set $D$ can be defined as a set of labeled sequences (i.e., program traces) $\{(Z_i, c_i) | Z_i \in \Sigma^*, c_i \in \{0, 1\}\}$, where $Z_i = z_1, z_2, z_3, \ldots, z_l$ is an input sequence and $c_i$ is a corresponding class label denoting 0 for “normal” label and 1 for “intrusion” label. Given the data set $D$, the goal of a learning algorithm is to find an intrusion detector $h : \Sigma^* \rightarrow \{0, 1\}$ that maximizes given criteria such as accuracy, F-1 measure, detection rate, and false positive rate.

If a probabilistic model is applied for the intrusion detector $h$ (e.g., Naive Bayes), then the probabilistic model $P_h$ specifies for a sequence $Z$ the probability $P_h(Z = z_1, z_2, z_3, \ldots, z_l)$ as follows.

1. For each class $c_i$, estimate the probabilities $P_h(c_i)$ using all the sequences $Z$ coupled with $c_i$.

2. For a new sequence $Z$, assign the class $c$ such that

$$c_h = \arg\max_{c \in \{0, 1\}} P_h(Z = z_1, z_2, z_3, \ldots, z_l | c) \cdot P_h(c).$$  \hspace{0.5cm} (1)

2.1. Naive Bayes Classifier. One of the important assumptions in the Naive Bayes classifier as a host-based intrusion detector is that each system call of the sequence is independent of the other system calls given the class label. Therefore, as for Naive Bayes, the classification (shown in (1)) of a new sequence will be formulated as follows:

$$c_{NB} = \arg\max_{c \in \{0, 1\}} P_h(c) \cdot \prod_{i} P_h(z_i | c).$$  \hspace{0.5cm} (2)

When the Naive Bayes classifier is applied to text or protein sequence classification, it treats each document or protein sequence as a bag or set of words or letters that denotes amino acid [9, 13]. There are a few researches [15–17] that explore intrusion detection tasks with a bag or set of system calls, but most intrusion detection researches focus on the $n$-gram approach [4, 14, 18, 19].

2.2. Naive Bayes with $n$-gram Features (NB $n$-gram). Since it is difficult to deal with variable length sequences directly, each sequence $Z \in \Sigma^*$ is mapped into a finite $n$-dimensional feature vector (i.e., $n$-gram features). In host-based intrusion detection tasks, if we want to monitor a program’s behavior, we consider the program’s trace as a sequence. Thus, to generate $n$-gram features from an input trace, a sliding window of size $n$ is applied to the trace, moving from the beginning of the trace to the end of the trace by one system call at each step, to generate a bag of $n$-gram features from the trace. Then, the probabilistic model for this $n$-gram representation is straightforward from (2):

$$c_{NBn\text{-gram}} = \arg\max_{c \in \{0, 1\}} \prod_{i=1}^{l-n+1} P_h(z_i, \ldots, z_{i+n-1} | c),$$  \hspace{0.5cm} (3)

where $l$ is the length of the sequence.

There is one serious problem in the NB $n$-gram approach. When $n$-gram features are generated from an original trace using a sliding window, one system call in the trace can be considered and included as many as $n$ times in the resulting $n$-gram features. This systematically violates the independence assumption of the Naive Bayes learning algorithm.

2.3. Interdependency Models of $n$-Grams (IM($n$)). To overcome the previously mentioned problem, we applied interdependency models of $n$-grams [8, 10] for scalable $n$-gram-based intrusion detection. The applied method tried to explicitly model the dependencies among the elements inside an $n$-gram feature generated from a sequence.

Figure 1 shows the model that describes dependencies among six consecutive elements in a sequence. Following the Junction Tree Theorem [20], the probabilistic model for IM($n$) is as follows:

$$c_{IM(n)} = \arg\max_{c \in \{0, 1\}} \prod_{i=1}^{l-2n+1} P_h(z_i, \ldots, z_{i+n-1} | c) / \prod_{i=2}^{l-n+1} P_h(z_i, \ldots, z_{i+n-2} | c) \cdot P_h(c).$$  \hspace{0.5cm} (4)

From Figure 1 and (4), it can be seen that the probabilistic graphical model of IM($n$) is a Markov Network where the probabilistic distribution is obtained by dividing the product of the marginals of the maximal cliques (maximally connected subgraphs) in the graph by the product of the marginals of the separators (overlaps among cliques).

Algorithm 1 shows the pseudocode of the intrusion detection algorithm using the interdependency model.
Algorithm 1: Intrusion detection algorithm with interdependency model.

\begin{algorithm}
\caption{Intrusion Detector($S$):}
\begin{algorithmic}
\STATE (1) \textbf{Input}: sequence data set $S = s_1, \ldots, s_n$ and interdependency model $\alpha$ as a probabilistic model.
\STATE (2) \textbf{Learning}: For each class $c_j$, estimate probabilities $P_\alpha(S = s_1, \ldots, s_n)$ of $c_j$ based on $D$ that comprises the intrusion detector $h$.
\STATE (3) \textbf{Testing}: For a novel sequence $\tilde{S} = s_1, \ldots, s_n$, predict the classification $\epsilon(\tilde{S})$ as follows:
\[ \epsilon(\tilde{S}) = \arg\max_{c_j \in C} \left\{ P_\alpha(\tilde{S} = s_1, \ldots, s_n | c_j) P(c_j) \right\} \]
\end{algorithmic}
\end{algorithm}

2.4. Support Vector Machines with n-Gram Features (SVM n-Gram). For the comparison of IM($n$)'s performance with the other data mining algorithm, we consider Support Vector Machines with $n$-grams as input.

It is of interest to compare IM($n$) and NB $n$-gram with SVM $n$-gram, because, in contrast to Naive Bayes with $n$-gram features (NB $n$-gram), SVMs do not rely on the independence assumption between features.

However, the SVM algorithm suffers from the explosion of the number of features as $n$ increases, because it takes at least $O(n^2)$ time and space to prepare a kernel matrix for SVM. In other words, since the SVM learning algorithm has nonlinear asymptotic complexity and due to the nature of $n$-gram attributes, as $n$ increases, the number of attributes and the size of the working memory increase drastically (i.e., curse of dimensionality). In the actual experiment, because of these computational and memory requirements, we were only able to conduct the experiment of SVM $n$-gram for $n = 1, 2, \text{and } 3$.

2.5. k-Truncated Suffix Tree. A suffix tree is a data structure to index a string \cite{11}. Figure 2 shows an example suffix tree for a string “banana$\$”, where “$\$” denotes the end of the string. The number in each node represents the number of pattern occurrences. For example, in the string “banana$\$”, “a” occurs three times and “na” occurs twice.

When the length of a string is $l$, then it takes $O(l)$ time \cite{21} to build a suffix tree for the string. Once a suffix tree is generated, then it takes $O(m)$ time to find a pattern string with length $m$. Also, with edge-label compression, it only needs $O(l)$ space for a suffix tree.

In practice, to store multiple strings, a generalized suffix tree is used. A generalized suffix tree is a storage that contains
all suffixes of a set of strings [11]. An example of a generalized suffix tree for the strings “bagle$$_1$$” and “beagle$$_2$$” is shown in Figure 4.

Even with the suffix tree storage, it still takes a lot of memory to save the entire traces of system calls. However, in our application, it is not necessary to store a whole trace into a suffix tree. Instead, we store $$n$$-gram features into the generalized suffix tree, as shown in Figure 3, which are of interest for generating intrusion detectors.

3. Experimental Setup and Results

To evaluate the performance of interdependency models of $$n$$-grams (IM($$n$$)), we compared its performance with Naive Bayes (NB), Naive Bayes with $$n$$-gram features (NB $$n$$-gram), and Support Vector Machines with $$n$$-gram features (SVM $$n$$-gram). For the experiment, we chose publicly available data sets from the University of New Mexico (UNM) [14] and MIT Lincoln Lab 1998 DARPA Intrusion Detection Evaluation Data Sets (MIT LL 1998) [22].

3.1. Data Sets

3.1.1. UNM Data Sets. The University of New Mexico (UNM) provides a number of system call data sets. Each data set corresponds to a specific attack or exploit. The data sets we tested are “live lpr”, “live lpr MIT”, and “denial of service” (DoS).

In UNM system call traces, each trace is an output of one program. Sometimes, one trace has multiple processes. In such cases, we have made as many sequences as the number of processes in the original trace. Thus, multiple sequences of system calls are made from one trace if the input trace has multiple processes in it. However, most traces have only one process and usually one sequence is created for each trace. Table 1 shows the number of original traces and the number of sequences for each data set.

There are two different mapping files in the UNM call traces we used for the experiment. One is Sun (live lpr and live lpr.MIT), and the other is Linux (denial of service). There are old and new Sun mapping files but only one system call is added to the new mapping file, so both can be easily

Table 1: The number of instances for each type of attack.

| Attack                  | Positive | Negative | Total |
|-------------------------|----------|----------|-------|
| Live lpr (1,lpr)        | 1001     | 1231     | 2232  |
| Live lpr MIT (1,lpr .MIT) | 1001     | 2703     | 3704  |
| Denial of service (atide) | 105      | 13726    | 13831 |
Table 2: Comparison of accuracy (A) and false positive rate (FP) of intrusion detectors generated by IM(n), NB n-gram, and SVM n-gram on the UNM live lpr data. The accuracies and false positive rates were estimated using 10-fold cross-validation. We calculated a 99% confidence interval on the accuracies and false positive rates. Note that when \( n \geq 3 \), SVM n-gram is infeasible because of computational and memory requirements.

| \( N \) | IM(n) A | IM(n) FP | NB n-gram A | NB n-gram FP | SVM n-gram A | SVM n-gram FP |
|-------|---------|---------|-------------|-------------|-------------|-------------|
| 1     | 84.09 ± 0.02 | 28.84 ± 0.02 | 84.09 ± 0.02 | 28.84 ± 0.02 | 100.00 ± 0.00 | 0.00 ± 0.00 |
| 2     | 99.78 ± 0.00 | 0.41 ± 0.00 | 98.30 ± 0.01 | 3.09 ± 0.01 | 99.96 ± 0.00 | 0.00 ± 0.00 |
| 3     | 99.96 ± 0.00 | 0.00 ± 0.00 | 99.01 ± 0.01 | 1.79 ± 0.01 | N/A | N/A |
| 4     | 99.96 ± 0.00 | 0.00 ± 0.00 | 99.60 ± 0.00 | 0.73 ± 0.00 | N/A | N/A |
| 5     | 99.96 ± 0.00 | 0.00 ± 0.00 | 99.82 ± 0.00 | 0.32 ± 0.00 | N/A | N/A |
| 6–8   | 100.00 ± 0.00 | 0.00 ± 0.00 | 99.87 ± 0.00 | 0.24 ± 0.00 | N/A | N/A |
| 9–10  | 99.96 ± 0.00 | 0.08 ± 0.00 | 99.82 ± 0.00 | 0.32 ± 0.00 | N/A | N/A |
| 11–20 | 99.96 ± 0.00 | 0.08 ± 0.00 | 99.78 ± 0.00 | 0.41 ± 0.00 | N/A | N/A |

3.2. Results. We applied the following performance measures:

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN},
\]

\[
\text{False Positive Rate} = \frac{FP}{FP + TN},
\]

where TP is the number of true positives, FP is the number of false positives, TN is the number of true negatives, and FN is the number of false negatives.

Table 2 shows accuracies and false positive rates of the three algorithms (IM(n), NB n-gram, and SVM n-gram) on the “UNM live lpr” data. IM(n) shows the best performance when \( n \) is from 6 to 8. Its performance is comparable to that of Support Vector Machines (SVMs). Overall, IM(n) shows better performance than NB n-gram.

As for the “UNM live lpr” data, IM(n) and NB n-gram show the best performance when \( n \) is from 6 to 8. The best accuracy and false positive rate of IM(n) are 100.00 and 0.00, and the best accuracy and false positive rate of NB n-gram are 99.87 and 0.24. These results correspond to the widely known claim that “six is the magic number” [23]. However, for the other data sets (“UNM live lpr MIT” and “UNM denial of service”) we have tested, the claim does not always hold.

Table 3 shows the results of the three algorithms for UNM live lpr MIT data. When \( n \) is from 17 to 20, IM(n) shows the best performance (accuracy is 99.97 and false positive rate is 0.00) over SVM (accuracy is 99.95 and false positive rate is 0.00) and NB n-gram (accuracy is 93.49 and false positive rate is 8.92).

In Table 4, which is for the UNM denial of service data, we can see that SVM with 2-grams shows the best performance (accuracy is 99.99 and false positive rate is 0.00), but the difference from the results of IM(n) is very marginal (accuracy is 99.93 and false positive rate is 0.00). Moreover, during the training stage, SVM learns its discriminative model by the Sequential Minimal Optimization algorithm [24], which takes more than several hours. In contrast, IM(n) takes only about one or two minutes and its performance is comparable to that of SVM.
Table 3: Comparison of accuracy (A) and false positive rate (FP) of intrusion detectors generated by IM($n$), NB $n$-gram, and SVM $n$-gram on UNM live for MIT data. The accuracies and false positive rates were estimated using 10-fold cross-validation. We calculated a 99% confidence interval on the accuracies and false positive rates. Note that when $n \geq 3$, SVM $n$-gram is infeasible because of computational and memory requirements.

| N   | IM($n$)     |     | NB $n$-gram |     | SVM $n$-gram |     |
|-----|-------------|-----|-------------|-----|-------------|-----|
|     | A           | FP  | A           | FP  | A           | FP  |
| 1   | 54.56 ± 0.02| 62.26 ± 0.02| 54.56 ± 0.02| 62.26 ± 0.02| 99.83 ± 0.00| 0.14 ± 0.00|
| 2   | 96.76 ± 0.01| 4.44 ± 0.01| 78.75 ± 0.02| 29.12 ± 0.02| 99.95 ± 0.00| 0.00 ± 0.00|
| 3   | 99.95 ± 0.00| 0.00 ± 0.00| 86.23 ± 0.01| 18.87 ± 0.02| N/A        | N/A  |
| 4   | 99.95 ± 0.00| 0.00 ± 0.00| 89.28 ± 0.01| 14.69 ± 0.02| N/A        | N/A  |
| 5   | 99.95 ± 0.00| 0.00 ± 0.00| 91.17 ± 0.01| 12.10 ± 0.01| N/A        | N/A  |
| 6   | 99.95 ± 0.00| 0.00 ± 0.00| 91.77 ± 0.01| 11.28 ± 0.01| N/A        | N/A  |
| 7   | 99.95 ± 0.00| 0.00 ± 0.00| 91.85 ± 0.01| 11.17 ± 0.01| N/A        | N/A  |
| 8   | 99.95 ± 0.00| 0.00 ± 0.00| 91.98 ± 0.01| 10.99 ± 0.01| N/A        | N/A  |
| 9   | 99.95 ± 0.00| 0.00 ± 0.00| 92.09 ± 0.01| 10.84 ± 0.01| N/A        | N/A  |
| 10  | 99.95 ± 0.00| 0.00 ± 0.00| 92.04 ± 0.01| 10.91 ± 0.01| N/A        | N/A  |
| 11  | 99.95 ± 0.00| 0.00 ± 0.00| 92.33 ± 0.01| 10.51 ± 0.01| N/A        | N/A  |
| 12  | 99.95 ± 0.00| 0.00 ± 0.00| 92.68 ± 0.01| 10.03 ± 0.01| N/A        | N/A  |
| 13  | 99.95 ± 0.00| 0.00 ± 0.00| 92.85 ± 0.01| 9.80 ± 0.01 | N/A        | N/A  |
| 14  | 99.95 ± 0.00| 0.00 ± 0.00| 92.95 ± 0.01| 9.66 ± 0.01 | N/A        | N/A  |
| 15  | 99.95 ± 0.00| 0.00 ± 0.00| 93.17 ± 0.01| 9.36 ± 0.01 | N/A        | N/A  |
| 16  | 99.95 ± 0.00| 0.00 ± 0.00| 93.28 ± 0.01| 9.21 ± 0.01 | N/A        | N/A  |
| 17  | 99.97 ± 0.00| 0.00 ± 0.00| 93.30 ± 0.01| 9.17 ± 0.01 | N/A        | N/A  |
| 18  | 99.97 ± 0.00| 0.00 ± 0.00| 93.36 ± 0.01| 9.10 ± 0.01 | N/A        | N/A  |
| 19-20| 99.97 ± 0.00| 0.00 ± 0.00| 93.49 ± 0.01| 8.92 ± 0.01 | N/A        | N/A  |

Table 4: Comparison of accuracy (A) and false positive rate (FP) of intrusion detectors generated by IM($n$), NB $n$-gram, and SVM $n$-gram on the UNM denial of service data. The accuracies and false positive rates were estimated using 10-fold cross-validation. We calculated a 99% confidence interval on the accuracies and false positive rates. Note that when $n \geq 3$, SVM $n$-gram is infeasible because of computational and memory requirements.

| N   | IM($n$)     |     | NB $n$-gram |     | SVM $n$-gram |     |
|-----|-------------|-----|-------------|-----|-------------|-----|
|     | A           | FP  | A           | FP  | A           | FP  |
| 1   | 98.69 ± 0.00| 0.92 ± 0.00| 98.69 ± 0.00| 0.92 ± 0.00| 99.98 ± 0.00| 0.01 ± 0.00|
| 2   | 99.27 ± 0.00| 0.01 ± 0.00| 99.24 ± 0.00| 0.04 ± 0.00| 99.99 ± 0.00| 0.00 ± 0.00|
| 3   | 99.24 ± 0.00| 0.01 ± 0.00| 99.06 ± 0.00| 0.69 ± 0.00 | N/A        | N/A  |
| 4   | 99.23 ± 0.00| 0.01 ± 0.00| 99.18 ± 0.00| 0.68 ± 0.00 | N/A        | N/A  |
| 5   | 99.23 ± 0.00| 0.01 ± 0.00| 99.24 ± 0.00| 0.65 ± 0.00 | N/A        | N/A  |
| 6   | 99.23 ± 0.00| 0.01 ± 0.00| 99.32 ± 0.00| 0.63 ± 0.00 | N/A        | N/A  |
| 7   | 99.24 ± 0.00| 0.01 ± 0.00| 99.40 ± 0.00| 0.59 ± 0.00 | N/A        | N/A  |
| 8   | 99.25 ± 0.00| 0.00 ± 0.00| 99.52 ± 0.00| 0.47 ± 0.00 | N/A        | N/A  |
| 9   | 99.36 ± 0.00| 0.00 ± 0.00| 99.66 ± 0.00| 0.33 ± 0.00 | N/A        | N/A  |
| 10  | 99.65 ± 0.00| 0.00 ± 0.00| 99.66 ± 0.00| 0.33 ± 0.00 | N/A        | N/A  |
| 11  | 99.80 ± 0.00| 0.00 ± 0.00| 99.66 ± 0.00| 0.33 ± 0.00 | N/A        | N/A  |
| 12  | 99.84 ± 0.00| 0.00 ± 0.00| 99.66 ± 0.00| 0.33 ± 0.00 | N/A        | N/A  |
| 13  | 99.87 ± 0.00| 0.00 ± 0.00| 99.66 ± 0.00| 0.33 ± 0.00 | N/A        | N/A  |
| 14  | 99.88 ± 0.00| 0.00 ± 0.00| 99.67 ± 0.00| 0.31 ± 0.00 | N/A        | N/A  |
| 15  | 99.89 ± 0.00| 0.00 ± 0.00| 99.67 ± 0.00| 0.31 ± 0.00 | N/A        | N/A  |
| 16  | 99.89 ± 0.00| 0.00 ± 0.00| 99.68 ± 0.00| 0.31 ± 0.00 | N/A        | N/A  |
| 17  | 99.91 ± 0.00| 0.00 ± 0.00| 99.69 ± 0.00| 0.30 ± 0.00 | N/A        | N/A  |
| 18  | 99.92 ± 0.00| 0.00 ± 0.00| 99.69 ± 0.00| 0.30 ± 0.00 | N/A        | N/A  |
| 19-20| 99.93 ± 0.00| 0.00 ± 0.00| 99.69 ± 0.00| 0.30 ± 0.00 | N/A        | N/A  |
Table 5: Comparison of accuracy (A) and false positive rate (FP) of intrusion detectors generated by IM(n), NB n-gram, and SVM n-gram on MIT LL 1998 data. We followed the train/test setting specified in the data set description. Note that when n ≥ 4, SVM n-gram is infeasible because of computational and memory requirements.

| N    | IM(n) | NB n-gram | SVM n-gram |
|------|-------|-----------|------------|
|      | A     | FP        | A          | FP        | A          | FP        |
| 1    | 36.57 | 62.49     | 36.57      | 62.49     | 81.55      | 7.45      |
| 2    | 80.33 | 0.06      | 80.36      | 0.03      | 80.53      | 8.28      |
| 3    | 80.35 | 0.03      | 80.35      | 0.05      | 80.28      | 9.35      |
| 4    | 80.37 | 0.00      | 80.34      | 0.03      | 80.53      | 8.28      |
| 5    | 80.33 | 0.05      | 80.35      | 0.03      | N/A        | N/A       |
| 6    | 80.33 | 0.05      | 80.30      | 0.08      | N/A        | N/A       |
| 7    | 80.37 | 0.04      | 78.50      | 2.89      | N/A        | N/A       |
| 8    | 79.51 | 1.11      | 78.45      | 2.95      | N/A        | N/A       |
| 9    | 79.44 | 1.20      | 75.22      | 6.93      | N/A        | N/A       |
| 10   | 75.99 | 5.50      | 75.12      | 7.07      | N/A        | N/A       |
| 11   | 75.93 | 5.57      | 74.99      | 7.27      | N/A        | N/A       |
| 12   | 75.93 | 5.57      | 75.05      | 7.29      | N/A        | N/A       |
| 13   | 75.93 | 5.58      | 74.85      | 7.56      | N/A        | N/A       |
| 14   | 75.91 | 5.59      | 71.11      | 12.84     | N/A        | N/A       |
| 15   | 75.79 | 5.76      | 65.10      | 25.30     | N/A        | N/A       |
| 16   | 76.97 | 6.26      | 52.47      | 42.21     | N/A        | N/A       |
| 17   | 72.23 | 12.46     | 50.39      | 44.82     | N/A        | N/A       |
| 18   | 71.98 | 12.67     | 50.42      | 44.77     | N/A        | N/A       |
| 19   | 71.99 | 12.66     | 50.36      | 44.88     | N/A        | N/A       |
| 20   | 74.95 | 9.06      | 49.88      | 44.93     | N/A        | N/A       |

In Table 5, we show the results of training and testing experiments on the MIT LL 1998 data sets. In these training and testing experiments, we use predefined training data sets of the MIT LL 1998 data sets for learning intrusion detectors and use predefined testing data sets for evaluating the intrusion detectors. Notice that we do not calculate confidence intervals because it is infeasible to obtain distributions from the train/test setting.

It can be seen that SVM with 1-gram shows the best performance on the accuracy with 81.55, but the false positive rate is 7.45. The difference from the results of IM(4) is marginal (accuracy is 80.37 and false positive rate is 0.00). Moreover, the false positive rate of IM(4) is far better than that of SVM with 1-gram. As discussed, the total learning and testing time of SVM is more than several hours, while, in contrast, it takes about one or two minutes for all the IM(n) models and their performances in general are comparable to those of SVMs.

In Figure 5, we show the Receiver Operating Characteristic (ROC) curves of train/test experiments on the MIT LL 1998 data sets. Area under the curve (AUC) of IM(4) is 0.7290, AUC of NB 2-gram is 0.6654, and AUC of SVM 1-gram is 0.7043. Thus, in terms of ROC, IM(4) outperforms SVM 1-gram.

In Table 6, we show the results of 10-fold cross-validation on the MIT LL 1998 data sets. It can be seen that SVM with 3-grams shows the best performance with the accuracy 93.34 and the false positive rate 4.40. Again, the difference from the results of IM(n) is marginal (accuracy is 93.24 and false positive rate is 3.93). Moreover, the running time of SVM 3-grams is more than a week for learning and testing, whereas the running time of IM(n) is less than two minutes and the performance of IM(n) is comparable to that of SVM.

In Figure 6, we show the Receiver Operating Characteristic (ROC) curves of train/test experiments on the MIT LL...
1998 data sets. Area Under the Curve (AUC) of IM(10) is 0.7742, AUC of NB 2-gram is 0.5271, and AUC of SVM 1-gram is 0.8729. Thus, in terms of ROC, SVM 1-gram shows the best performance among the three.

To show the big picture of comparing IM(\(n\)) and NB \(n\)-gram, in Figure 7, we show ROC curves of the train/test experiments on the MIT LL 1998 data sets, when \(2 \leq n \leq 17\).

4. Conclusion

4.1. Related Work. Peng and Schuurmans [8] introduced \(n\)-gram augmented Naive Bayes and applied their algorithms to text classification. Silvescu et al. [10] proposed interelement dependency models which are similar to \(n\)-gram-augmented Naive Bayes. However, to the best of our knowledge, there has
Table 6: Comparison of accuracy (A) and false positive rate (FP) of intrusion detectors generated by IM(n), NB n-gram, and SVM n-gram on MIT LL 1998 data. The accuracies and false positive rates were estimated using 10-fold cross-validation. We calculated a 99% confidence interval on the accuracies and false positive rates. Note that when $n \geq 3$, SVM n-gram is infeasible because of computational and memory requirements.

| $N$ | IM(n) | NB n-gram | SVM n-gram |
|-----|-------|-----------|-----------|
|     | A     | FP        | A         | FP        | A         | FP        |
| 1   | 32.55 ± 0.37 | 88.55 ± 0.25 | 32.55 ± 0.37 | 88.55 ± 0.25 | 91.83 ± 0.21 | 0.03 ± 0.01 |
| 2   | 74.54 ± 0.34 | 26.17 ± 0.34 | 60.61 ± 0.38 | 41.91 ± 0.38 | 92.45 ± 0.21 | 4.46 ± 0.16 |
| 3   | 48.33 ± 0.39 | 56.18 ± 0.39 | 46.97 ± 0.39 | 57.44 ± 0.39 | 93.34 ± 0.19 | 4.40 ± 0.16 |
| 4   | 85.39 ± 0.28 | 13.84 ± 0.27 | 50.73 ± 0.39 | 53.29 ± 0.39 | N/A | N/A |
| 5   | 92.17 ± 0.21 | 6.02 ± 0.19 | 56.28 ± 0.39 | 47.08 ± 0.39 | N/A | N/A |
| 6   | 93.12 ± 0.20 | 4.97 ± 0.17 | 60.77 ± 0.38 | 42.02 ± 0.38 | N/A | N/A |
| 7   | 93.17 ± 0.20 | 3.97 ± 0.15 | 76.68 ± 0.33 | 24.54 ± 0.33 | N/A | N/A |
| 8   | 93.23 ± 0.20 | 3.96 ± 0.15 | 77.58 ± 0.33 | 22.43 ± 0.33 | N/A | N/A |
| 9   | 93.22 ± 0.20 | 3.94 ± 0.15 | 78.74 ± 0.32 | 21.18 ± 0.32 | N/A | N/A |
| 10  | 93.24 ± 0.20 | 3.93 ± 0.15 | 79.33 ± 0.32 | 20.53 ± 0.32 | N/A | N/A |
| 11  | 93.24 ± 0.20 | 3.94 ± 0.15 | 78.71 ± 0.32 | 21.92 ± 0.32 | N/A | N/A |
| 12  | 93.37 ± 0.19 | 4.00 ± 0.15 | 78.64 ± 0.32 | 21.54 ± 0.32 | N/A | N/A |
| 13  | 91.65 ± 0.22 | 1.44 ± 0.09 | 82.43 ± 0.30 | 12.55 ± 0.26 | N/A | N/A |
| 14  | 91.39 ± 0.22 | 1.73 ± 0.10 | 86.21 ± 0.27 | 8.12 ± 0.21 | N/A | N/A |
| 15  | 91.35 ± 0.22 | 1.77 ± 0.10 | 85.09 ± 0.28 | 14.54 ± 0.27 | N/A | N/A |
| 16  | 91.27 ± 0.22 | 2.36 ± 0.12 | 85.10 ± 0.28 | 13.92 ± 0.27 | N/A | N/A |
| 17  | 91.50 ± 0.22 | 5.39 ± 0.18 | 85.24 ± 0.28 | 13.20 ± 0.26 | N/A | N/A |
| 18  | 91.42 ± 0.22 | 4.98 ± 0.17 | 85.37 ± 0.28 | 13.03 ± 0.26 | N/A | N/A |
| 19  | 91.47 ± 0.22 | 5.41 ± 0.18 | 85.54 ± 0.27 | 12.83 ± 0.26 | N/A | N/A |
| 20  | 91.46 ± 0.22 | 5.43 ± 0.18 | 85.68 ± 0.27 | 13.87 ± 0.27 | N/A | N/A |

been no research on application of interelement dependency models for intrusion detection tasks.

Rieck and Laskov [4] used language models to detect unknown network attacks. They used a trie data structure [25, 26] to compute the similarity between two traces. The generalized suffix tree we used is more advantageous for storing n-gram features in that it does not take constant time to find a substring in a trie data structure. We performed our experiments with $n$ ranging from 1 to 20, whereas Rieck and Laskov did the experiments when $n$ is only 1, 3, and 5.

Most intrusion detection research focuses on the $n$-gram approach [4, 14, 18, 19]. However, there have been a few research efforts [15–17] that explore intrusion detection tasks with a bag or set of system calls. Liao and Vemuri [15] applied the K-Nearest Neighbor text classification method to intrusion detection tasks. Kang et al. [16] proposed a bag of system calls representation for intrusion detection. They performed various experiments for misuse detection and anomaly detection and showed that a bag of system calls representation is effective for misuse detection. Liu et al. [17] compared different system call feature representations and concluded that system call alone is not sufficient for detecting insider threats. Our work extends the representation to $n$-gram and shows that the resulting intrusion detectors are more accurate than $n$-gram features in terms of accuracy and false positive rate.

Forrest et al. [18] devised a Sequence Time-Delay Embedding (STIDE) intrusion detector which takes the $n$-gram approach with a few thresholds. Tan and Maxion [23] tried to define the operational limit of STIDE and concluded that 6-gram is enough for anomaly detection. Our work shows that the interelement dependency model (IM(n)) is more accurate than the $n$-gram approach and, for some attack types (UNM live lpr MIT and UNM denial of service), six is not always the magic number.

4.2. Summary and Future Work. We discussed an application of interelement dependency models to $n$-grams stored in a $k$-truncated generalized suffix tree ($k$-TGST) directly to classify intrusive sequences. We evaluated the performance of our method with those of Naive Bayes and Support Vector Machines (SVMs) with $n$-gram features by the experiments on intrusion detection benchmark data sets.

Experimental results on the University of New Mexico (UNM) benchmark data sets and MIT Lincoln Lab 1998 DARPA intrusion detection evaluation data sets show that our method, which solves the problem of independence violation that happens when $n$-gram-features are directly applied to Naive Bayes (i.e., Naive Bayes with $n$-gram features), yields intrusion detectors with higher accuracy than those from Naive Bayes with $n$-gram features and shows comparable accuracy to those from SVM with $n$-gram features.

For scalable and lightweight counting of $n$-gram features, we use the $k$-truncated generalized suffix tree mechanism for storing $n$-gram features. With this mechanism, we tested the performance of the classifiers up to 20-gram in our experiment, which illustrates the scalability and accuracy
of \( n \)-gram augmented Naive Bayes with the \( k \)-truncated generalised suffix tree storage mechanism.

As for future work, we plan to apply \( n \)-gram representation to system call arguments [27], because system call arguments are important for accurate intrusion detection. Since there has been a lot of research on \( n \)-gram approaches, it is of interest to devise an \( n \)-gram representation where each element is the audit record data structure (i.e., system call and its arguments). However, there have been no researches we are aware of for the application of \( n \)-gram representation for the audit record data structure.

References

[1] E. Charniak, Statistical Language Learning, MIT Press, Cambridge, MA, USA, 1994.

[2] W. Lee, S. J. Stolfo, and K. W. Mok, “A data mining framework for building intrusion detection models,” in Proceedings of the IEEE Symposium on Security and Privacy, pp. 120–132, May 1999.

[3] A. Murali and M. Rao, “A survey on intrusion detection approaches,” in Proceedings of the 1st International Conference on Information and Communication Technology (ICICT ’05), pp. 233–240, August 2005.

[4] K. Rieck and P. Laskov, “Detecting unknown network attacks using language models,” in Proceedings of the 3rd International Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA ’06), pp. 74–90, Berlin, Germany, 2006.

[5] M. Z. Shafiq, S. A. Khayam, and M. Farooq, “Embedded malware detection using Markov \( n \)-grams,” in Proceedings of the 5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA ’08), pp. 88–107, Paris, France, 2008.

[6] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “Training algorithm for optimal margin classifiers,” in Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, pp. 144–152, ACM Press, New York, NY, USA, July 1992.

[7] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, NY, USA, 1995.

[8] F. Peng and D. Schuurmans, “Combining naive Bayes and \( n \)-gram language models for text classification,” in Advances in Information Retrieval, F. Sebastiani, Ed., vol. 2633 of Lecture Notes in Computer Science, pp. 335–350, New York, NY, USA, 2003.

[9] C. Andorf, A. Silvescu, D. Dobbs, and V. Honavar, “Learning classifiers for assigning protein sequences to gene ontology functional families,” in Proceedings of the 5th International Conference on Knowledge Based Computer Systems (KBCS ’04), pp. 256–265, 2004.

[10] A. Silvescu, C. Andorf, D. Dobbs, and V. Honavar, “Inter-element dependency models for sequence classification,” Tech. Rep., Iowa State University, 2004.

[11] D. Gusfield, Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, Cambridge University Press, New York, NY, USA, 1st edition, 1997.

[12] J. C. Na and K. Park, “Data compression with truncated suffix trees,” in Proceedings of the Data Compression Conference (DDC ’00), Snowbird, UT, USA, March 2000.

[13] T. M. Mitchell, Machine Learning, McGraw-Hill, New York, NY, USA, 1997.

[14] C. Warrender, S. Forrest, and B. Pearlmuter, “Detecting intrusions using system calls: alternative data models,” in Proceedings of the IEEE Symposium on Security and Privacy, pp. 133–145, May 1999.

[15] Y. Liao and V. R. Vemuri, “Using text categorization techniques for intrusion detection,” in Proceedings of the 11th USENIX Security Symposium, pp. 51–59, USENIX Association, Berkeley, CA, USA, 2002.

[16] D.-K. Kang, D. Fuller, and V. Honavar, “Learning classifiers for misuse and anomaly detection using a bag of system calls representation,” in Proceedings of the 6th Annual IEEE System, Man and Cybernetics Information Assurance Workshop (SMC ’05), pp. 118–125, West Point, NY, USA, June 2005.

[17] A. Liu, C. Martin, T. Hetherington, and S. Matzner, “A comparison of system call feature representations for insider threat detection,” in Proceedings of the 6th Annual IEEE System, Man and Cybernetics Information Assurance Workshop (SMC ’05), pp. 340–347, West Point, NY, USA, June 2005.

[18] S. Forrest, L. Allen, A. S. Perelson, and R. Cherukuri, “Self-nonself discrimination in a computer,” in Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, pp. 202–212, May 1994.

[19] W. Lee and S. Stolfo, “Data mining approaches for intrusion detection,” in Proceedings of the 7th USENIX Security Symposium, pp. 79–94, San Antonio, TX, USA, 1998.

[20] R. G. Cowell, S. L. Lauritzen, A. P. David, D. J. Spiegelhalter, and D. J. Spiegelhater, Probabilistic Networks and Expert Systems, Springer, Secaucus, NJ, USA, 1999.

[21] E. Ukkonen, “On-line construction of suffix trees,” Algorithmica, vol. 14, no. 3, pp. 249–260, 1995.

[22] R. Lippmann, D. Fried, I. Graf et al., “Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation,” in Proceedings of the DARPA Information Survivability Conference and Exposition, IEEE Computer Society Press, Los Alamitos, CA, USA, 2000.

[23] K. M. C. Tan and R. A. Maxion, “Why 6? Defining the operational limits of stide, an anomaly-based intrusion detector,” in Proceedings of the Symposium on Security and Privacy, pp. 188–201, May 2002.

[24] Platt and J. C. C., “Fast training of support vector machines using sequential minimal optimization,” Advances in Kernel Methods, pp. 185–208, 1999.

[25] R. de la Briandais, “File searching using variable length keys,” in Proceedings of the AFIPS West Joint Computer Conference, vol. 15, pp. 295–298, 1959.

[26] E. Fredkin, “Trie memory,” Communications of the ACM, vol. 3, no. 9, pp. 490–499, 1960.

[27] D. Mutz, F. Valeur, G. Vigna, and C. Kruegel, “Anomalous system call detection,” ACM Transactions on Information and System Security, vol. 9, no. 1, pp. 61–93, 2006.
