Intrusion Trace Classification using Inter-element Dependency Models with k-Truncated Generalized Suffix Tree[1]

Dae-Ki Kang* and Pilsung Kang** [2]

** Division of Computer & Information Engineering, Dongseo University*
47, Jurye-Ro, Sasang-Ku, Busan, 617-716, Korea
dkkang@dongseo.ac.kr
*** Flash Solution Development Team, Memory Division*
Samsung Electronics, Korea
pils.kang@samsung.com

Abstract

We present a scalable and accurate method for classifying program traces to detect system intrusion attempts. By employing inter-element dependency models to overcome the independence violation problem inherent in the Naïve Bayes learners, our method yields intrusion detectors with better accuracy. For efficient 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 to scale up the classifiers, which cannot be easily achieved by SVM (Support Vector Machine) based methods that require implausible computing power and resources for accuracy.

Keywords: *Inter-element Dependency Models, k-Truncated Generalized Suffix Trees, Intrusion Detection*

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[1,2]. For example, in 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 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 [3] has been widely used for featurization of system call sequences[4-7].

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

1. **Curse of Dimensionality:** Since the number of distinct system calls are 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

[1] This paper is an extended version of 2012 International Conference on Information Science and Technology. However, there have been significant additions from the conference version.
[2] Corresponding author: Mobile: +82 10 6231 8326; tel.: +82 31 325 8451 (office).

have non-linear space complexity (such as Support Vector Machines (SVM)[8,9]) 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 [10,11].
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 inter-element dependency models [12,10] of n-gram features to intrusion detection tasks and compared their performance with those 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 [13,14] 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.

2. Method

First of all, we introduce Naive Bayes with n-gram features (NB n-gram), Inter-Dependency 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 $\Sigma = \{s_1, s_2, s_3, \dots, s_m\}$ 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) $\{\langle Z_i, c_i\rangle | Z_i \epsilon \Sigma^*, c_i \epsilon \{0,1\}\}$ where $Z_i = z_1, z_2, z_3, \dots, z_l$ 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 $h: \Sigma^* \longrightarrow \{0,1\}$ that maximizes given criteria such as accuracy, F-1 measure, detection rate, false positive rate, etc.

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 $P_h(Z_i = z_1, z_2, z_3, \dots, z_l)$ as follows:

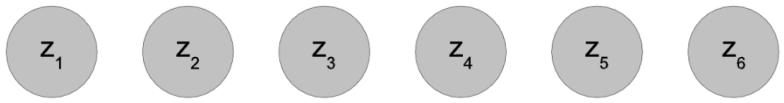
1. For each class c_i, estimate the probabilities $P_h(c_i)$ using all the sequences Z coupled with c_i.
2. For a new sequence Z, assign the class c such that

$$c_h = \text{argmax}_{c\epsilon\{0,1\}} P_h(Z_i = z_1, z_2, z_3, \dots, z_l | c) P_h(c)$$

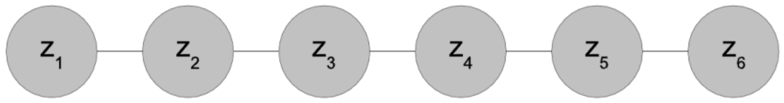
2.1. Inter-Dependency Models of n-Grams (IM(n))

To overcome the previously mentioned problem, we applied Inter-Dependency Models of n-Grams [12,10] 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.

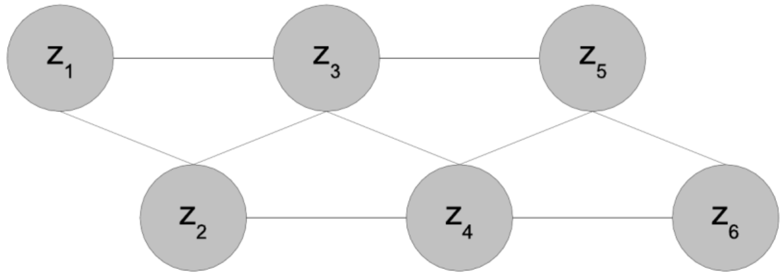
Figure 1 shows the model that describes dependencies among six consecutive elements in a sequence.



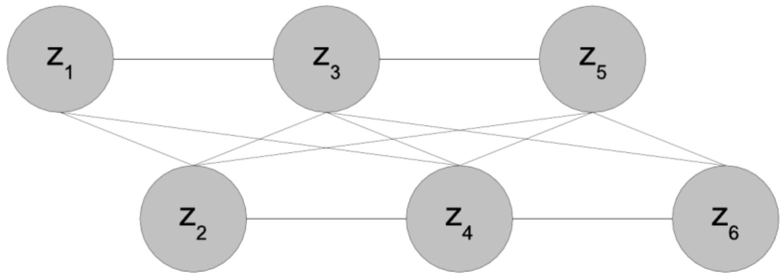
(a) IM(1)=Naive Bayes



(b) IM(2) depicts a dependency between an element and its direct successor



(c) IM(3) depicts dependencies among an element, its 1st successor, and its 2nd successor



(d) IM(4) depicts dependencies among an element and its three closest successors

Figure 1. Graphical Models that Incorporate the Dependencies among the Six Consecutive Elements in a Sequence

Following the Junction Tree Theorem [15], the probabilistic model for IM(n) is as follows:

$$c_{IM(n)} = \operatorname{argmax}_{c\epsilon\{0,1\}} \frac{\prod_1^{l-n+1} P_h(Z_i = z_1, z_2, z_3, \dots, z_{i-n+1}|c)}{\prod_2^{l-n+1} P_h(Z_i = z_1, z_2, z_3, \dots, z_{i-n+1}|c)} P_h(c) \qquad (1)$$

From Fig. 1 and Equation 1, 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).

Figure 2 shows the pseudo-code of the intrusion detection algorithm using inter-dependency model.

```
IntrusionDetector(S):
begin
    1. Input : sequence data set $S = s_1, s_2, s_3, \ldots, s_n$ and inter-dependency model $\alpha$ as a
       probabilistic model.
    2. Learning : For each class $c_j$, estimate probabilities $P_\alpha(S = s_1, s_2, s_3, \ldots, s_n)$ of
       $\alpha(c_j)$ based on D that comprises the intrusion detector h.
    3. Testing : For a novel sequence $\hat{S} = s_1, s_2, s_3, \ldots, s_n$, predict the classification $c(\hat{S})$
       as follows:
           $c(\hat{S}) = \mathrm{argmax}_{c_j \epsilon C}\{P_\alpha(\hat{S} = s_1, s_2, s_3, \ldots, s_n | c_j) P(c_j)\}$
end.
```

Figure 2. Intrusion Detection Algorithm with Inter-dependency Model

2.2. k-Truncated Suffix Tree

A suffix tree is a data structure to index a string[13]. When the length of a string is l, then it takes O(l) time[16] 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[13].

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, which are of interest for generating intrusion detectors.

3. Conclusion

We discussed an application of inter-element 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 (SVM) with n-gram features by the experiments on intrusion detection benchmark data sets.

For scalable and efficient 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 generalized suffix tree storage mechanism.

Acknowledgements

This research was supported by Dongseo University's Ubiquitous Appliance Regional Innovation Center research grants from Ministry of Knowledge Economy of the Korean government. (No. B0008352).

References

[1] J. T. Kim, J. H. Kho, M. S. Hong, C. W. Son, B. Park, D. W. Lee and G. Lee, "A study on intrusion protection techniques against Linux kernel backdoor", The Journal of IWIT (The Institute of Webcasting, Internet and Telecommunication) 9 **(2009)** pp. 201-207.

[2] S. J. Oh, "Design and evaluation of a weighted intrusion detection method for VANETs", The Journal of IWIT (The Institute of Webcasting, Internet and Telecommunication) 11 **(2011)** pp. 181-188.

[3] E. Charniak, "Statistical Language Learning", MIT Press, Cambridge, MA, USA **(1994)**.

[4] W. Lee, S. J. Stolfo and K. W. Mok, "A data mining framework for building intrusion detection models", In: IEEE Symposium on Security and Privacy. **(1999)** pp. 120-132.

[5] A. Murali and M. Rao, "A survey on intrusion detection approaches", In: 1st International Conference on Information and Communication Technologies (ICICT 2005). **(2005)** pp. 233-240.

[6] K. Rieck, P. Laskov, "Detecting unknown network attacks using language models", In: Proceedings of 3rd International Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2006), Berlin, Germany **(2006)** pp. 74-90.

[7] 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 2008) **(2008)**.

[8] B. E. Boser, I. M. Guyon and V. N. Vapnik, "A training algorithm for optimal margin classifiers", In: COLT '92: Proceedings of the 5th annual workshop on Computational learning theory, New York, NY, USA, ACM Press **(1992)** pp. 144-152.

[9] V. N. Vapnik, "The nature of statistical learning theory", Springer-Verlag New York, Inc., New York, NY, USA **(1995)**.

[10] F. Peng and Schuurmans, "Combining naive Bayes and n-gram language models for text classification", In Sebastiani, F., ed.: Advances in Information Retrieval, 25th European Conference on IR Research (ECIR 2003). Volume 2633 of Lecture Notes in Computer Science, Springer **(2003)** pp. 335-350.

[11] 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 2004). **(2004)** pp. 256-265.

[12] A. Silvescu, C. Andorf, D. Dobbs and V. Honavar, "Inter-element dependency models for sequence classification", Technical report, Iowa State University **(2004)**.

[13] D. Gusfield, "Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology", 1st ed. Cambridge University Press **(1997)**.

[14] J. C. Na and K. Park, "Data compression with truncated suffix trees", In: Proceedings of Data Compression Conference 2000. **(2000)** pp. 565.

[15] R. G. Cowell, S. L. Lauritzen, A. P. David, D. J. Spiegelhalter and D. J. Spiegelhater, "Probabilistic Networks and Expert Systems", Springer-Verlag New York, Inc., Secaucus, NJ, USA **(1999)**.

[16] E. Ukkonen, "On-line construction of suffix-trees", Algorithmica 14 **(1995)** pp. 249-260.

Authors

Dae-Ki Kang is an assistant professor at Dongseo University in South Korea. He was a senior member of engineering staff at the attached Institute of Electronics & Telecommunications Research Institute in South Korea. He earned a PhD in computer science from Iowa State University in 2006. His research interests include social network services, machine learning, relational learning, statistical graphical models, metaheuristics, ontology learning, Tower of Hanoi, multimedia systems, intrusion detection, Web firewall, and computer vision.

Prior to joining Iowa State, he worked at a Bay-area startup company and at Electronics and Telecommunication Research Institute in South Korea. He received a science master degree in computer science at

Sogang University in 1994 and a bachelor of engineering (BE) degree in computer science and engineering at Hanyang University in 1992.

Pilsung Kang is a senior engineer at Samsung Electronics, where he develops embedded software for solid-state drives. His research interests include computational science, parallel computing, data mining, and embedded systems. Kang has a PhD in computer science from Virginia Tech.