Improved Intrusive Process Detection Via Text Categorization Of System Call Sequences

Kurt Buehner

Abstract

This paper compares the efficacy of two anomaly detection classifiers with respect to the classification of processes as either intrusive or non-intrusive. To the task of process classification, both classifiers treat processes as system call sequences, encode those system call sequences as text documents, and apply the k-nearest neighbor text categorization method to classify the processes. In these text documents, system call attributes are represented as weighted frequencies. One of the classifiers, an established anomaly detection classifier encodes system calls associated with a process as a vector of weighted frequencies. The other, putative classifier encodes pairs of sequential system calls as a two-dimensional table of weighted frequencies. For a process, each entry in the table represents the weighted frequency of a transition from one system call to the next. As such the putative classifier categorizes processes based on local system call ordering and system call frequency, while the established classifier relies on system call frequency alone. Using the DARPA Off-line Intrusion Detection Evaluation data, this paper shows that the putative, weighted-table anomaly detection classifier more accurately identifies intrusive processes than the weighted-vector k-nearest neighbor classifier. By employing a more detailed encoding of system call sequences, the putative, weighted-table classifier achieves a higher detection rate while maintaining a low false positive rate relative to the weighted-vector classifier.

1. Introduction

For Host-based Intrusion Detection Systems (IDS), two classes of algorithms have emerged; misuse detection and anomaly detection. Misuse detection relies upon the signatures of known intrusion attempts (such as malformed packets) to identify preemptively known intrusion attempts[1]. However, novel attacks can bypass an IDS that relies upon misuse detection alone[2]. Anomaly detection compensates for the shortcomings of misuse detection by reactively identifying intrusive activity. Thereby, anomaly detection complements misuse detection with respect to IDS algorithms.

Anomaly detection relies upon statistical analysis of system characteristics (system calls, performance, user profiles, network traffic) to detect abnormal system behavior corresponding to intrusion. Anomaly detection has been used to detect intrusion and initiate appropriate counter-measures; notification, intruder tracking, server lock-down, and port of entry identification. There exists no 100% effective anomaly detection system, and fresh intrusion attempts plague online computer systems. Thus, security analysts seek newer, more effective anomaly detection algorithms.

A process profile is a numerical representation of process attributes. For the purpose of comparison, anomaly detection classifiers employ process profiles to characterize runtime instances of programs. Some effective process profiles have been created from patterns that arise in system call sequences. An intruder may attempt to gain control of a system by corrupting or gaining control of a process. Once corrupted, a process' behavior differs from that program's normal runtime behavior. Using the principle that corrupted processes deviate from normal, non-corrupt behavior; Liao et al. successfully created an anomaly detection classifier, the weighted-vector classifier[3]. Specifically, Liao et al. established an analogy between text categorization and intrusion detection. To this end, Liao et al. created process profiles based on system call sequences, treated the process profiles as text documents, and applied the k-nearest neighbor (k-NN) text categorization method to classify these process profiles.

There are two characteristic metrics for assessing IDS effectiveness: detection rate and false alarm rate[4][5]. The detection rate measures the rate of correct attack-identification by an IDS during a particular environment and time frame. An IDS's detection rate will vary with the set of attacks used in a test.

The false alarm rate is a measure of false positives produced by an IDS in a given environment during a particular time frame. The false alarm rate is computed as the quotient of the number of false positives and the total number of normal events in a time period. In the context of IDS's, a normal event refers to any non-intrusive TCP/IP session or any non-intrusive process. With regards to an IDS, a false positive (or rather a false alarm) is an alert caused by normal, non-malicious traffic that results from a misclassification[5]. To compare the effectiveness of two IDS's by false alarm rates; one must compute false alarm rates for each IDS using data collected from the same environment during the same time frame. The need for such rigor arises from the volatility and unpredictability of real network traffic. As such, the DARPA off-line intrusion detection evaluation[6] was created to provide a collection of data against which IDS's could be evaluated. Furthermore, the DARPA off-line intrusion detection evaluation data was chosen to judge the effectiveness of Liao et al.'s anomaly classifier.

A receiver operating characteristic (ROC) curve plots percent detection rate versus percent false positive rate for an IDS. For the points on an ROC curve, the IDS must use the same configuration during testing for false positives and attack detections. Such curves allow for graphical comparison of IDS's.

To categorize a process as normal or anomalous, Liao et al. proposed a classifier that compared vectors of weighted system call frequencies. For a suspect/test process, Liao et al.'s weighted-vector classifier computed the cosine-similarities of
that test-vector and each pre-selected, normal-training-vectors for the test program. The 'k' largest similarity scores were extracted via the k-nearest neighbor algorithm, and the mean of the 'k' largest similarity scores was computed. If that average was above a pre-chosen threshold, then the weighted-vector classifier categorized the process as normal. Otherwise the process was classified as abnormal. Here, Liao et al. assumes that abnormal process behavior arises from intrusion attempts.

Liao et al. used the weighted-vector classifier along with a few other criteria to construct and test an intrusion detection algorithm (Figure-1). The additional criteria involved the selection of 50 system calls (Table-1) that occurred in the 1998 DARPA Basic Security Module (BSM)[7][8] audit data. If a process used a system call not in the set of 50 system calls, then Liao et al.'s weighted-vector algorithm automatically categorized that process as anomalous. Liao et al.'s extra criteria adds a condition not integral to the weighted-vector classifier itself. The effects of the extra criteria are discussed below. None the less, Liao et al. provided an analogy between text categorization and anomaly detection, that proved efficacious with respect to intrusive process detection.

Table 1:
List of 50 distinct, monitored system calls.

<table>
<thead>
<tr>
<th>access</th>
<th>fchdir</th>
<th>login</th>
<th>pipe</th>
<th>setpgid</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit</td>
<td>fchown</td>
<td>logout</td>
<td>putmsg</td>
<td>setrlimit</td>
</tr>
<tr>
<td>audion</td>
<td>fcntl</td>
<td>lstat</td>
<td>readlink</td>
<td>setuid</td>
</tr>
<tr>
<td>chdir</td>
<td>fork</td>
<td>memcntl</td>
<td>rename</td>
<td>stat</td>
</tr>
<tr>
<td>chmod</td>
<td>fork1</td>
<td>mkdir</td>
<td>mmap</td>
<td>setaudit</td>
</tr>
<tr>
<td>chown</td>
<td>getaudit</td>
<td>mmap</td>
<td>setaudit</td>
<td>su</td>
</tr>
<tr>
<td>close</td>
<td>getmsg</td>
<td>munmap</td>
<td>setegid</td>
<td>sysinfo</td>
</tr>
<tr>
<td>creat</td>
<td>ioctl</td>
<td>nice</td>
<td>seteuid</td>
<td>unlink</td>
</tr>
<tr>
<td>execve</td>
<td>kill</td>
<td>open</td>
<td>setgids</td>
<td>utime</td>
</tr>
<tr>
<td>exit</td>
<td>link</td>
<td>pathdonf</td>
<td>setgroups</td>
<td>vfork</td>
</tr>
</tbody>
</table>

2. Methodology
2.1. Process Classification.

To categorize a suspect process as either normal or intrusive, the K-NN classifier calculates the similarity between that process profile and each training process profile, and uses the class labels of the K-nearest neighbors to predict the class of the new process.

As a run-time instance of program, processes periodically and predictably request services from the operating system in the form of system calls. Depending upon the degree of determinism in a program's control flow and the consistency of requests made of a program, patterns arise in the sequence of system calls for a process. For example, programs with stateful behavior exhibit a high level of consistency with regards to the system call sequences produced at runtime. A high degree of determinism in a program will produce a high degree of consistency with regards to the system call sequences produced at runtime. Furthermore, for even the most non-deterministic programs at runtime, the sequence of system calls will vary along with the variability of the requests or inputs. In other words, given the same input, a program's processes will typically produce the same results and follow the same control flow to generate those results. On these bases, patterns arise in system call sequences.

Given the above assumptions, under normal usage the sequences of system calls produced by a program at runtime are specific to that program, finite in number, and predictable. As such, a set of ordered system calls sequences characterizes a program's normal runtime behavior.

Text categorization is the process of grouping text documents into one or more predefined categories based on document content (words). Text categorization techniques form an assemblage of machine learning techniques as based on statistical classification. By encoding system call sequences as text documents, one can exploit text categorization techniques for system call sequence classification. Processes that belong to the same class will cluster together in the vector space. Each cluster will represent a group of similar system call sequences for a process. Table-2 further explains the analogy between process profiles and text documents with regard to k-nearest neighbor text categorization method. Essentially, each process is treated as a sequence of system calls. As per Liao et al., each system call sequence is analogous to a text document in which each system call represents a word.

2.2. Term Weighting Methods.

Equation-1 and Equation-2 define two weighting methods by which to compute, $d_i$ the weight of the i-th word in the j-th training document. Equation-1 computes the $d_i$ by the frequency weighting method, and Equation-2 computes $d_i$ by the term-frequency inverse-document-frequency ($\text{tf} \cdot \text{idf}$) method. Here, $c_{ij}$ is the number of occurrences of the i-th word in the j-th training document; $C_j$ is the total number of system calls in the j-th training document. N is the number of documents in the training set; $f_i$ is the frequency of the i-th word in the j-th training document; $M$ is the number of distinct words in the training documents; $n_i$ is the number of training documents in which the i-th word occurs.

$$d_y = f_j = c_{ij} / C_j \quad \text{(Equation-1)}$$

$$d_y = \sqrt{\sum_i f_i \cdot \log \frac{N}{n_i}} \quad \text{(Equation-2)}$$

$$\text{sim}(X, D_j) = \frac{\sum_{i \in X \cap D_j} x_i \cdot d_i}{\|X\| \cdot \|D_j\|} \quad \text{(Equation-3)}$$

Equation-3 specifies the calculation for cosine similarity. In equation-3, X is the test document, D_j is the j-th training document; $x_i$ is the weight of the i-th word in document X, $d_i$ is the weight of the i-th word in document D_j. $\|X\|$ is the weight of norm of X; and $\|D_j\|$ is the norm of D_j.

In Boolean weighting the weight of a distinct system call is assigned the value of 1 if it occurs in a process and 0 otherwise. Frequency weighting discriminates between system calls based on number of occurrences in a process, and accounts for variability of system call sequence length. Here, weight is proportional to frequency. Like frequency weighting, $\text{tf} \cdot \text{idf}$ weighting accounts for frequency of occurrence, and accounts for document length variability. In addition, $\text{tf} \cdot \text{idf}$ places more weight on terms that occur in fewer documents and accounts for the frequency of a term though out the entire training document collection.
3. Experiment

Experiments were performed to establish the following hypothesis: the weighted-table classifier provides superior anomalous process detection to that attained by Liao et al.'s weighted-vector classifier. For both classifiers, anomaly classifier effectiveness was measured by two metrics, attack detection rate and false positive rate.

The testing of the hypothesis necessitated an experiment in three phases: reproduce the results of Liao et al.'s weighted-vector algorithm on the 1998 DARPA Basic Security Module data[3][6][8], compare the effectiveness of the two classifiers on the 1998 DARPA BSM data, and compare the effectiveness of the two classifiers on the 1999 DARPA BSM data. To ensure that the experiments were conducted in a manner consistent the Liao et al.[3]; Liao et al.'s weighted-vector algorithm was reproduced and their data set recreated from the 1998 DARPA off-line intrusion detection evaluation data.

Abstractly, the computation of the detection rate and false positive rate involved four steps; 1) generate the attack-free process profiles for the training documents, 2) generate the process profiles for test documents, 3) use an anomaly detection classifier to compare the test process profile and training process profiles for the same apparent program name, and 4) classify the test process as intrusive on non-intrusive.

To establish validity and reproducibility, two iterations of the experiment were performed. One iteration was performed on the 1998 DARPA BSM audit data and the other was perform on the 1999 DARPA BSM data. From the 1998 DARPA off-line intrusion detection evaluation data, Liao et al.'s data set was recreated. The 1998 DARPA BSM data included seven weeks of training data with labeled attacks. In the seven weeks there were five day in which no attacks appear in the BSM audit data. The BSM data from four of the attack free days (week3-wed, week2-tue, week7-tue, week7-thr) was used to create the training process profiles, while the remaining day (week7-wed) was used for determining the false positive rate. From the BSM audit the following information was extracted programatically for each process; ordered list of system calls, program name, date, and start time of the process. The extracted information was programatically converted into the process profiles for the weighted-vector algorithm and the weighted-table algorithm.

For the weighted-table classifier, the system call sequences were encoded differently than for the weighted-vector classifier. In the weighted-table process profile, system call sequences were encoded as the number of times system-call, followed system-call. In the weighted-vector process profile, the number of occurrences of system call, was encoded. Both encodings contained sufficient information to compute Boolean weight, frequency weight, and the tf•idf weight of each entry in the table or vector. From these weights, one can compute the cosine-similarity for two process profiles. Thus allowing quantitative measurement of similarity between a test process and a training process.

The duplicate process profiles were identified and removed from the training data sets. Duplicates were identified as those processes which shared the same program name and system call sequence. In this fashion, a set of unique non-attack process profiles was composed for the training document set.

<table>
<thead>
<tr>
<th>Table 2: Analogous observables for text catagorization and weighted-table classifier.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Algebraic</th>
<th>Text Catagorization Term</th>
<th>Weighted Table Intrusion Detection Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Number of training documents</td>
<td>Number of training process profiles.</td>
</tr>
<tr>
<td>M</td>
<td>Number of distinct words in training set</td>
<td>Number of distinct system call to system call traisitions</td>
</tr>
<tr>
<td>n_i</td>
<td>Number of documents in which i-th word occurs</td>
<td>Number of process profiles in which i-th transition occurs</td>
</tr>
<tr>
<td>c_i</td>
<td>Number of occurrences of i-th word in the j-th document</td>
<td>Number of occurrences of i-th transition occurs</td>
</tr>
<tr>
<td>C_j</td>
<td>Total number of words in the j-th document</td>
<td>Total number of transitions in the j-th process profile</td>
</tr>
<tr>
<td>f_i</td>
<td>Frequency of the i-th word in the j-th training document</td>
<td>Frequency of the i-th transition in the j-th training process</td>
</tr>
<tr>
<td>D_j</td>
<td>The j-th training document</td>
<td>The j-th process profile</td>
</tr>
<tr>
<td>X</td>
<td>The test document</td>
<td>The test process profile</td>
</tr>
<tr>
<td>d_i</td>
<td>The weight of the i-th word in the j-th training document</td>
<td>The weight of the i-th transition in the j-th process profile</td>
</tr>
</tbody>
</table>
For the experiment involving the 1999 DARPA BSM audit data, the above operational procedures were repeated with the exception of the audit log choice for the test data and for the training data. The process profiles for the training data were extracted from the five days of week-1's BSM audit logs. The test data was extracted from the five days of week-4's BSM audit logs. The processes associated with attacks were identified. Using week-1's unique process profiles as a training set and using week-4 as the test set, each process in week-4 was classified by both classifiers. From the results of the classification, the detection rate and false positive rates were computed with varying values of k (number of nearest neighbors) and the similarity-score-threshold.

![Figure-2: ROC curves for Liao et al's k-NN algorithm with tf•idf weighting method.](image)

```
build the training/normal data set, D={ D_j }
for each process X in the test data do
    for each process D_j in the training set do
        compute sim( X, D_j );
        if sim( X, D_j ) equals 1.0 then
            X is normal; exit;
        find k biggest scores of sim( X, D_j );
        calculate sim_avg for k-nearest neighbors;
        if sim_avg is greater than threshold then
            X is normal;
        else
            X is abnormal;
```

Figure-3: Pseudo code for anomaly detection classifier only.

4. Results

Both the anomaly detection classifiers use the same operational definition for an anomalous process; any process whose system call sequence does not exactly match a system call sequence in the training set for that program, and whose average cosine-similarity for k-nearest neighbors falls below a certain threshold. So the operational variables under consideration are threshold, number of nearest neighbors, choice of weighting method, thoroughness of the training documents, and nature of the attacks themselves.

In the BSM audit data, there were duplicate sessions for two exploits, eject and warezclient. Just as was done by Liao et al., these duplicate instances of the eject-exploit and the warezclient-exploit were correctly classified, but excluded from the 1998 test data. Excluding these duplicate sessions from the seven week 1998 DARPA training set, there were 35 TCP/IP sessions that were both intrusive and appeared in the 1998 BSM audit logs. These 35 intrusive sessions generated 66 intrusive processes from which to compute the attack detection rate. The BSM audit data for Wednesday of week-7 contained 5285 normal processes from which the false positive rate was computed.

Using the weighted-vector and weighted-table classifiers alone, the attack rates and false positive rates were computed from the 1998 DARPA BSM training data. For varying threshold values (0.0, 0.40, 0.60, 0.66, 0.70, and 0.72) and the tf•idf weighting method; Figure-4, Figure-5, and Figure-6 depicts the ROC curves for k=5, k=10, and k=25 respectively. Liao et al.‘s algorithm contains an ancillary criteria; automatically classify any process with an "unknown system call" as anomalous. Liao et al.’s designation of unknown system call appears arbitrary. The pseudo code of Liao et al.’s algorithm appears in Figure-1. Figure-3 displays the pseudo code pertinent only to the anomaly classifiers themselves. When one removes the ancillary criteria from Figure-1’s pseudo code, one arrives at the algorithm depicted in Figure-3. To maintain consistency with Liao et al.'s procedures; in Figure-2, Figure-4, Figure-5, and Figure-6, the reported data for the weighted vector algorithm was computed with Liao et al.’s ancillary criteria. Figure-3 depicts the pseudo code used to compute all data reported for the putative, weighted-table classifier.

For varying threshold values (0.0, 0.40, 0.60, 0.66, 0.70, and 0.72), Liao et al. reported the effectiveness of an algorithm based on the weighted vector classifier. For purposes of validation, Liao et al.’s effort was successfully reproduced in Figure-2. Figure-2 shows the ROC curve for Liao et al.’s anomaly detection algorithm on the 1998 DARPA BSM data. Liao et al.’s algorithm was reproduced from the procedures outlined in their experiment[3].

For the 1998 DARPA BSM data, Figures-7 depicts the ROC curve for the weighted-table classifier and the weighted-vector classifier (without the ancillary criteria). Figure-7 plots the ROC curves for the classifiers on the 1999 DARPA BSM data for tf•idf weighting, k=25 neighbors, and threshold=0.72. As per Liao et al., these are the optimal parameters for their classifier. As such these parameters provide the most favorable detection rate with an acceptable false positive rate for the weighted-vector classifier. Hence all comparisons of the two classifiers are made at the optimal values reported by Liao et al.

By eliminating Liao et al.’s ancillary criteria, the weighted-vector’s false positive rate remains unchanged, but the detection rate decreased to 95 %. The attack rate reduction reflects three misclassified (false negative) instances of the attacks, satan. All three misclassified attacks possessed weighted-vector process profiles with high similarity scores relative to the training process profiles for the program, finger. However, even without the ancillary criteria, the weighted-table classifier properly categorized these attacks. With respect to the weighted-table classifier, Liao et al.’s ancillary criteria has no effect upon the achieved detection rate and false positive rate. For the 1998 BSM audit data, Table-3 enumerates the mean of the greatest 25 similarity scores associated with attack-processes (tf•idf weighting). The attacks listed in Table-3 represent a collection of user-to-root and remote to local attacks[4].
5. Discussion

Both classifiers achieved low false positive rates (false positive rate < 0.84%) for the test data. The low false positive rates implies that the training sets sufficiently covered the sample space. In the DARPA BSM data, a typical day's simulation included 5000 to 6000 processes about a third of which were non-redundant. As with real processes, the non-deterministic programs produced dissimilar process profiles while deterministic applications produced similar process profiles. Hence, the DARPA BSM data provides realistic process activity. This in turn allows false positive rates to be validly measured. The non-deterministic, low redundancy applications included user-shells (tcsh and sh), mail, and text editing processes. The minimal redundancy of these applications shows that random human influences were properly simulated in the DARPA data. On the other hand deterministic processes displayed minimal redundancy with respect to their process profiles. The low process profile redundancy rate implies that the DARPA data sets simulate realistic network traffic in a controlled test bed.

From the mean of the average similarity scores displayed in Table-3 for each attack, there exists a significant difference between the average similarity score achieved by the two classifiers. Where sufficient attack instances existed, the t-test was applied to the mean of the average similarity scores. As per the two-tailed t-test, the differences in the means similarity
scores does not arise from chance. Since the false positive rates remain nominal for the two classifiers, then the results of Table-3 support the hypothesis. Furthermore, Table-4 corroborates Liao et al.'s results[3] and the hypothesis. With respect to Table-5, the tf•idf weighting detection rates favor the weighted Table classifier. As such, experiments with the 1998 and 1999 DARPA data imply that the weighted-table classifier more accurately identifies anomalous processes than the weighted-vector classifier.

| Table 4: Comparison of attack detection rates with threshold \( \alpha = 0.72, k=25 \) for the 1998 DARPA BSM audit data and varying weighting methods. |
|---|---|---|---|
| classifier | boolean weighting | frequency weighting | tf•idf weighting |
| weighted vector | 0.80 | 0.80 | 0.95 |
| weighted table | 0.80 | 0.90 | 1.00 |

For the 1999 DARPA data, Table-5 displays the detection rates computed for the experiment with various weighting methods. For the 1998 and 1999 data sets, Boolean and frequency weighting methods appear inferior to tf•idf weighting. In the 1999 data set, the Boolean and frequency weighting techniques performed inadequately for anomaly intrusion detection. The tf•idf weighting method assigns term weight based largely on the differences between training process profiles. So such, the tf•idf weighting method assigned greater weight to a term that occurs in fewer documents in the training set. Herein lies tf•idf weighting's ability to correlate information about the entire training set and the reason for tf•idf weighting method's superiority. The low attack detection rates achieved with the 1999 data set for frequency and Boolean weighting arise from the high level of similarity between attack processes and normal processes. Any discrepancy between the attack detection rates shown in Table-4 and Table-5 arises from attack construction improvement between the 1998 and 1999 data sets.

6. Conclusion

The weighted-table classifier is not a panacea for anomaly detection. However, weighted-table classifier exhibits improvement over the weighted-vector classifier proposed by Liao et al. Using the 1998 and 1999 DARPA off-line intrusion detection evaluation data, the two classifiers were used to compute the cosine-similarity of the 'k' most similar processes. Based on these average similarity scores, the two classifiers were compared with respect to detection rate and false positive rate. The putative weighted-table classifier detected more anomalous processes at lower thresholds than the weighted-vector classifier. In the 1998 and 1999 DARPA Off-line data, the putative classifier detected 100% of the anomalous processes at a average-similarity threshold of 0.72 and \( k=25 \) nearest neighbors. The putative anomaly detection classifier has been shown to provide superior detection of anomalous/intrusive processes while maintaining low false positive rates ( < 0.84% ). The two classifiers displayed marginal difference with regards to false positive rates. The putative classifier has proven effective at classifying anomalous processes resulting from user-to-root attacks and remote-to-local attacks. As such, the putative classifier can reactively identify those novel, intrusive processes which bypass signature detection alone. Hence, the putative classifier has been shown to be a suitable for use in an anomaly detection algorithm.

7. References


