Computer Log Anomaly Detection Using Frequent Episodes

Perttu Halonen, Markus Miettinen, and Kimmo Hätönen

Abstract In this paper, we propose a set of algorithms to automate the detection of *anomalous frequent episodes*. The algorithms make use of the hierarchy and frequency of episodes present in an examined sequence of log data and in a history preceding it. The algorithms identify changes in a set of frequent episodes and their frequencies. We evaluate the algorithms and describe tests made using live computer system log data.

1 Introduction

In security analysis, knowledge-based intrusion detection tools are using pre-defined log entry patterns of known incidents. The patterns cover known vulnerabilities in the system. However, before a pattern can be created, one has to find the vulnerability and identify its traces. Unfortunately, in many cases the vulnerability is found only after it has been exploited.

To identify unknown attacks one should inspect all log entries and their contexts to classify them as a sign of normal operation or a possible intrusion. Due to the huge volume of data, it is impossible to make detailed analyses for everything. However, since an intrusion into a network typically causes chages in event logs, e.g., a missing or changed event in a usual pattern, one can reduce the set of entries requiring thorough inspection by focusing the analysis on only those entries that are anomalous with regard to earlier data. In this paper, we describe *anomaly detection* methods which can be used for this kind of pre-screening of security mon-

Perttu Halonen · Kimmo Hätönen
Nokia Siemens Networks, PL 6, FI-02022, Finland,
e-mail: perttu.halonen@nsn.com, kimmo.hatonen@nsn.com

Markus Miettinen
Nokia Research Center, Itämerenkatu 11–13, FI-00180 Helsinki, Finland,
e-mail: markus.miettinen@nokia.com

itoring data. Our problem domain is telecommunications network information security monitoring, but the algorithms we present are suitable also for other problem domains.

Related Work Several approaches have been proposed for detecting sudden changes in execution of computer software. Forrest et al. have presented an interesting immunology based approach [3] that imitates the biological immune system. Ko et al. have proposed specification based anomaly detection [4], in which one has to use a formal language to specify which execution traces of computer programs are allowed. Lane and Brodley have proposed to detect anomalous behaviour of a computer system user by monitoring her command history [5].

Organisation of this paper is as follows. Sect. 2 introduces mining of frequent closed episodes and presents our proposal for anomaly detection algorithms. In Sect. 3, we describe the tests we have performed with our algorithms, and in Sect. 4 analyse and discuss the results. Finally, Sect. 5 summarises this paper.

2 Frequent Episode Anomaly Detection Methods

Log sequences consist of *log entries*. A log entry *e* is a triplet (t,E,s) consisting of a *time t*, an *event type E* and a *source s*. *Frequent episodes* are collections of event types occurring frequently within a given time *w* in the entries of a log sequence [6].

The concept of frequent episodes is a derivative of *frequent sets* [1]. Frequent sets are sets of items that frequently occur together in the records of a database. The APRIORI algorithm for mining frequent sets [1] can be modified to compute frequent unordered episodes [7]. In this paper, we use unordered episodes.

To mine frequent episodes, we divide the log entry sequence into consecutive non-overlapping time windows of maximal width *w*. In addition, we require that as soon as a log entry with an event type equal to some event type already included in the window is encountered again, the current window is terminated and a new window started. Thus, each event type can occur only once within each time window.

We use a set of *closed frequent episodes* instead of the set of all frequent episodes. The *closure* of an episode is its largest super-episode that shares the same frequency, and a *closed episode* is a frequent episode that is equal to its closure. The set of all closed episodes effectively encodes information about all freqent episodes and can be used to simplify processing without loosing information about the occurrences of the frequent episodes. Closed frequent episodes are a derivative of so-called closed frequent sets [8, 2].

In the following, we present algorithms that can be used for identifying anomalies in frequent episodes in a set of analysed log data. The aim of the algorithms is to identify new or modified patterns from a set of analysed data.

Let $\mathscr{E}$ be the set $\{E_1,E_2,...,E_n\}$ of all possible event types E_i that appear in the log data. We denote with C the set of all closed frequent episodes that appear in the analysed log L, i.e. $C = \{f \subseteq \mathscr{E} \mid f \text{ is a closed frequent episode in } L\}$. For each

episode $f \in C$ we store as $f.freq$ the frequency of f in L. Frequency denotes here the absolute count of an episode's occurrences.

Finding Changes in Closed Episodes. Algorithm 1 identifies changes in frequent closed episodes by comparing the frequency of an episode $f \in C$ to the frequency of its *specialisations*, i.e., its super-episodes $p \in C$ such that $f \subset p$. If the difference in the frequencies is small, it could be an indication of the fact that a normal event sequence represented by the specialisation has changed. The changed event sequence does not contribute to the frequency of the specialisation, but it does contribute to the frequency of at least some of the subepisodes. A small difference in the episode frequencies can therefore be interpreted as an indication of such a change in an existing episode and can be reported as a potential anomaly.

Algorithm 1 Finding changes in closed episodes

C	Set of frequent closed episodes found in the analysed log L and their occurrence frequencies.
Δ_f	Threshold specifying maximum difference between the frequencies of compared episodes.
A	Set of superepisode-subepisode pairs whose frequencies in L differ at most by Δ_f.

for all $f \in C$ **do**
  for all $p \in C$ s.t. $f \subset p$ **do**
    if $f.freq - p.freq \leq \Delta_f$ **then**
      $A \leftarrow A \cup (p, f)$
return A

The algorithm has one parameter, Δ_f. It is the maximum difference of the frequency of the sub- and super-episodes for them to be considered an anomalous pair. The output of algorithm 1 can be used for analysing the input log and identifying those windows that are anomalous. Algorithm 2 below marks as anomalous those windows of the log data, which match to any of the episode pairs in the set A of anomalous episode pairs.

Algorithm 2 Marking log windows as anomalous

A	Set of anomalous superepisode-subepisode pairs
L	Log data to be marked
W	Set of anomalous windows in the log data

repeat
  $w \leftarrow getFirstWindow(L)$
  $L \leftarrow L \setminus w$
  if $\exists (p, f) \in A$ s.t. $f \subset w \wedge p \not\subset w$ **then**
    $w.anomaly_{body} \leftarrow f$
    $w.anomaly_{missing} \leftarrow p \setminus f$
    $W \leftarrow W \cup \{w\}$
until $L = \emptyset$ **return** W

Finding novel episode patterns. Algorithm 3 searches for new occurrences of frequent closed episodes, which are not present in the preceding history data. We

denote with P the episode profile that has been calculated based on a history H of log data. P constitutes the model of normal behaviour and it contains all closed frequent episodes f that appear in H, i.e. $P = \{f \subseteq \mathscr{E} \mid f \text{ is a closed frequent episode in } H\}$.

The algorithm compares the profile episodes in P with the closed frequent episodes C found from the analysed log data L. Such novel frequent episodes are potentially interesting because they may indicate completely new types of activity in the log data.

Algorithm 3 Finding Novel Episode Patterns

P Set of profile episodes extracted from log history database H.
C Set of closed frequent episodes found from the analysed log L.
N Set of new episodes appearing in C but not in the episode profile P.

$N \leftarrow C \setminus P$
for all $n \in N$ **do**
  if $\exists p \in P$ s.t. $n \subset p$ **then**
    $N \leftarrow N \setminus n$
return N

3 Tests

We tested the anomaly detection algorithms on several types of logs obtained from a telecommunications network, covering a continuous period of 42 days. The data were divided into data sequences and closed frequent episodes were mined for each data sequence, using a frequency threshold of 5 occurrences and limiting the maximum window length for episodes to 3600 seconds.

We wanted to know, how large a fraction of the input data would be considered anomalous by our methods. We first measured the relation between the log entries marked as anomalous and the total amount of log entries present in the analysed log. For each data sequence, we calculated the number of log entries in windows W covered by the set of anomalous episodes obtained from algorithm 2 which we divided by the total amount of log entries in the sequence.

Figure 1 shows the results of our tests on data from the application log and the system log of the network management system. The fraction of anomalous log entries for the system log varies between ca. 5% and 10%, whereas the fraction of anomalous log entries stays below 2% for the application log. One can see that with the exception of a few observations, the measures maintain the same order of magnitude within the same log type.

The second property we measured is the amount of novel episode patterns detected from the analysed log data. That is, the profile contained the frequent closed episodes that occurred in five days preceeding the analysed data sequence. We ex-

ecuted then algorithm 3 on the analysed data sequence and counted the amount of novel frequent episodes. The counts are show in Figure 2.

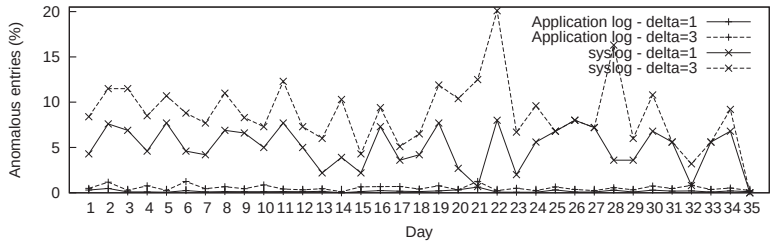

Fig. 1 Fraction of daily log entries marked as anomalous by algorithm 2. The number of daily entries in the application log was between 3284 and 5357 (average 5097). Reported anomalies varied between 0 and 31 (average 9) for $\Delta_f = 1$ and between 6 and 64 (average 28) for $\Delta_f = 3$. The system log contained 223 to 546 entries (average 280), for which 0 to 30 (average 14) anomalies were reported for $\Delta_f = 1$ and 0 to 75 (average 25) anomalies for $\Delta_f = 3$.

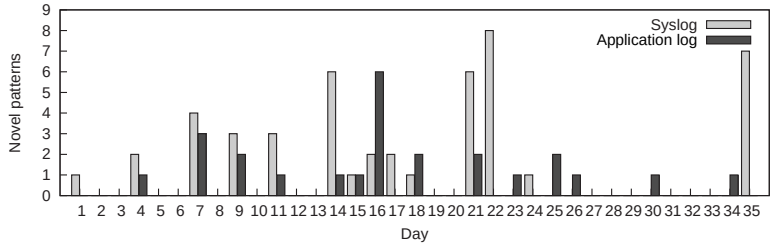

Fig. 2 Daily amounts of new episode patterns in the application and system logs

4 Discussion

Figure 1 shows a clear difference between the analysed log types. The application log contains a large amount of routinely recorded event records. The percentage of anomalous entries remains low due to the large overall record mass of the application log. The system log on the other hand monitors the operation of the basic system components and records any errors and deviations occurring in the system. The amount of log entries is smaller and the relative likelihood of error occurrences higher.

The novel episode pattern measure in Figure 2 shows that the appearance of entirely new episodes is rather exceptional for both shown log types. The number of

reported daily novel episodes remains so small that they could be easily inspected on a daily basis by a human monitoring officer.

The results suggest that algorithms 1 and 2 can be used to filter out log entries that deviate from the usual frequent behaviour. Such pre-filtering would enable an expert system or even human analysts to focus the subsequent analysis on log entries that are known to be anomalous with regard to the bulk of the data. The filtering seems to be more effective ($> 98\%$ in our application log example) for log types with higher entry volumes, where obviously abnormal activities do not dominate the data set. However, also for log types showing more volatile behaviour, significant data filtering efficiency can be achieved (ca. $90 - 95\%$ in our system log example).

5 Summary

In this paper, we have presented algorithms to detect *anomalous frequent episodes*. The algorithms make use of the hierarchy and frequency of episodes present in an examined log data sequence and in a history preceding it. The algorithms identify changes in a set of frequent episodes and their frequencies. We have evaluated the presented algorithms and described tests made using live network log data.

References

1. R. Agrawal et al. Fast discovery of association rules. In U.M. Fayyad et al., editors, *Adv. in knowl. discovery and data mining*, pages 307 – 328. AAAI, Menlo Park, CA, USA, 1996.
2. J. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proc. PAKDD'00*, volume 1805 of *LNAI*, pages 62–73, Kyoto, Japan, April 2000. Springer.
3. S. Forrest et al. Self-nonself discrimination in a computer. In *Proc. of the 1994 IEEE Symp. on Research in Security and Privacy, Los Alamos, CA*, pages 202–212. IEEE Computer Society Press, 1994.
4. C. Ko et al. Execution monitoring of security-critical programs in distributed systems: a specification-based approach. *1997 IEEE Symp. on Security and Privacy*, 00:175–187, 1997.
5. T. Lane and C.E. Brodley. Sequence matching and learning in anomaly detection for computer security. In *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, pages 43–49, July 1997.
6. H. Mannila et al. Discovering frequent episodes in sequences. In *Proc. of the First Int. Conf. on Knowledge Discovery and Data Mining (KDD'95)*, pages 210–215, Montreal, Canada, August 1995. AAAI Press.
7. H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In E. Simoudis et al., editors, *Proc. of the Second Int. Conf. on Knowledge Discovery and Data Mining (KDD'96)*, pages 146–151, Portland, Oregon, August 1996. AAAI Press.
8. N. Pasquier et al. Discovering frequent closed itemsets for association rules. *LNCS*, 1540:398–416, 1999.