Audit File Reduction Using N-Gram Models*

Fernando Godínez[1], Dieter Hutter[2], and Raúl Monroy[3]

[1] Centre for Intelligent Systems, ITESM–Monterrey,
Eugenio Garza Sada 2501, Monterrey, 64849, Mexico
`fgodinez@itesm.mx`

[2] DFKI, Saarbrücken University, Stuhlsatzenhausweg 3,
D-66123 Saarbrücken, Germany
`hutter@dfki.de`

[3] Department of Computer Science, ITESM–Estado de México,
Carr. Lago de Guadalupe, Km. 3.5, Estado de México,
52926, Mexico
`raulm@itesm.mx`

While some accurate, current Intrusion Detection Systems (IDS's) get rapidly overwhelmed with contemporary information workload [1, 2]. This problem partly dwells in the number of repetitive spurious information that IDS's unnecessarily analyse. Using this observation, we propose a methodology which can be used to significantly remove such spurious information and thus alleviate intrusion detection.

Throughout our experiments we have considered host-based intrusion detection, using the 1998 DARPA repository [3]. The IDS is thus assumed to make an audit from a set of sessions, each of which is a sequence of system calls and corresponds to either of the following services: `telnet`, `ftp`, `smtp`, `finger` or `echo`.

The reduction methodology is twofold:

1 **(Training):** We identify and tag with a new label the sequences of system calls of most frequent occurrence, considering only intrusion-free sessions; and

2 **(Reduction):** We shrink an input session replacing every such a repetitive sequence with its corresponding new label.

Folding repetitive sequences significantly reduces the length of a given session: we obtained an average reduction factor of 4 (3.6, worst case scenario, and 4.8, best case one.) It also helps intrusion detection: for example, it is much faster to build an hidden Markov model-based misuse IDS; and it slightly increases the detection ratio but, more importantly, the false positive ratio is only 1% higher.

Training. To identify sequences of system calls of most frequent occurrence, we use n-gram theory [4]. *N-gram theory* comprises a collection of probabilistic

* This research is supported by three research grants CONACyT 33337-A, CONACyT-DLR J200.324/2003 and ITESM CCEM-0302-05.

methods for estimating the probability that a sequence of symbols, i.e. system calls, will occur in a larger, unseen sequence, i.e. a session. Using such probabilities, we have selected a collection of sequences of system calls, henceforth called *n-grams*, that, when folded, are hoped to largely shrink an input session.

The training step consists of 4 steps: i) n-gram extraction; ii) n-gram priority assignment; iii) n-gram selection; and iv) n-gram overlapping avoidance. N-gram extraction consists of identifying all the n-grams arising throughout every session as well as counting their occurrences. Because it consists of one or more interleaved processes, each session is first manipulated so that it is turned into a sequence of orderly processes.

Priority assignment consists of associating with every possible n-gram an estimation as to how much will it reduce a given session, called *its priority.* This step poses two technical problems. First, some n-grams may not have turned up in any training session and yet they all must be assigned a priority. To overcome this problem, we use a *discounting strategy*, namely good-Turing, with which we can estimate an occurrence probability and then use it to compute a priority. Second, some n-grams yield a high reduction ratio regardless of the service, but others impact only on a specific service. For a service not to be neglected only because it has a less representative body in the training sessions, priority should account for both the estimated reduction factor per day, considering sessions of several services, and per service.

The priority of an n-gram, if appears in the training corpora, is given by:

$$Pr_t = \frac{n \times (f_t + 1)}{N_t} \tag{1}$$

where n is the size of the n-gram, N is the total number of system calls within a day considering either all services or a given one, f is the frequency of occurrence of the associated n-gram, and where t stands either for d, a day, or s, a given service. By contrast, the priority for an n-gram, if an probability of occurrence is estimated, is given by:

$$Pr_t = P_t \times n \tag{2}$$

In the third step, n-grams are first separated into two classes, depending on whether or not they occur in the training corpora, and then each class is manipulated as follows. First, each n-gram is put into 2 separate sets, in one the n-gram is labelled with Pr_d and with Pr_s in the other. Then, each set is arranged according to the n-grams priority in descending order. Then, using the 4 separate arrays, we select as many top n-grams as required in order to achieve a designated reduction factor. If sorting turns out a prohibitively expensive process, as is in the normal case of huge sessions, we suggest to depict histograms and then examine them seeking the n-grams with most promising reduction factor. In this case, it is helpful to consider n-grams whose frequency of occurrence is (nearly) a multiple of the number of different sessions. The rationale behind this selection criterion is that such n-grams are likely to have appeared along every session.

The fourth, final step is n-gram overlapping avoidance. N-grams tend to overlap with each other, they might intersect at some point. To avoid overlapping

mmap(2)||success	$\circ$	close(2)||success	$\circ$	open(2)_-_read||success	$\circ$
mmap(2)||success	$\circ$	mmap(2)||success	$\circ$	munmap(2)||success	$\circ$
mmap(2)||success	$\circ$	close(2)||success	$\circ$	open(2)_-_read||success	$\circ$
mmap(2)||success	$\circ$	mmap(2)||success	$\circ$	munmap(2)||success	$\circ$
mmap(2)||success	$\circ$	close(2)||success	$\circ$	open(2)_-_read||success	$\circ$
mmap(2)||success	$\circ$	mmap(2)||success	$\circ$	munmap(2)||success	$\circ$
mmap(2)||success	$\circ$	close(2)||success	$\circ$		

Fig. 1. An example reduction n-gram of size 20 with $Pr_d = 0.1135$. $\circ$ denotes the sequence constructor function

as well as using n-grams with a higher reduction ratio, we form a queue with the selected n-grams, ordering them by priority. Then, we create a window of a size equal to the largest n-gram in the queue. In production, this window is filled with an input session and then tested against the n-grams in the priority queue. By substituting n-grams with higher ratio we guarantee that, even if there is an overlapping, only the n-grams that provide maximum reduction are used. Notice that by substituting an n-gram with a new symbol we are avoiding further substitution on that segment resulting in overlapping elimination. We avoid substitution because the newly added symbol is not present in any n-gram used in the substitution.

We run this training methodology using 5 DARPA log files. We initially selected 200 n-grams from the 4 independent arrays. Our training experiments show that only 11 n-grams are really used out of the 100 n-grams selected from the occurrence-frequency, Pr_d array; only 5 n-grams are used out of the 50 ones extracted from the occurrence-probability Pr_d array; and only 3 n-grams were used from the 50 ones selected from the two Pr_d arrays. Thus 89% of 93% of the selected n-grams are overlapping. Since these n-grams do not overlap, any subsequent reductions do not consider a priority. Our main result is a set of 19 reduction n-grams that, as discussed below, provide an average reduction rate of 74%. One such an n-gram is shown in Fig. 1. The n-gram reduction set is available upon request, by sending e-mail to the 1st author.

Reduction. When tested against the training sessions, the n-gram reduction set provided an average reduction of 74%. Then, we validated the n-gram set by making it reduce a collection of unseen sessions, taken from 5 different DARPA log files from the 1999 repository. The results obtained from the validation experiments are shown in table 1; they portray an average reduction of 70.5%. Given that the training log files and the validation ones are from a different year, we conclude that the n-grams are general enough to fold sessions from different users.

Impact on Intrusion Detection. Using our folding approach, we have reduced up to 75% the length of a typical session. This reduction allows us to build and use hidden Markov models (HMM's) with larger sequences than those used in current approaches [1, 2, 5]. HMM's take a large amount of time for training.

Table 1. Validation Results

Log File	size(file)	size(reduced file)	Reduction factor	N-grams used
1	776,000	270,000	65.3%	7
2	1,800,000	486,000	73%	12
3	1,150,000	344,000	70.1%	5
4	801,000	175,000	78.2%	9
5	1,158,000	392,000	66.2%	5
Telnet	209,000	48,000	77.1%	5

Wagner and Soto, describe the disadvantages of using only short sequences as the detection base using HMM's [6]. We used entire sessions containing the attacks for both, our training data and detection testing. We used a single HMM for each attack.

All the attacks used throughout our experimentations are present in the 1998 and 1999 DARPA repositories. We obtained a detection ratio of 97%. False positive rate is 10.5%. This false positive rate is still high. By reducing the detection threshold, false positive ratio also lowers. Initially we used a 90% similarity measure, i.e., to be labelled as an attack, the tested sequence should be 90% similar to the training sequence. When increased to 95%, false positives were reduced to 6.5%. Detection ratio was also lowered to 92%.

By using reduced and a similarity measure of 90%, detection ratio increased to 98%, and false positive rate was 11.5%. By increasing the similarity measure to 95%, false positives lowered to 7% and the detection ratio also lowered to 94%. We tested the same attacks for reduced and non-reduced sessions. The difference in false positives was found in short attacks such as `eject`. Most of the false positives were normal sessions labelled as one of these short attacks.

From these results we can conclude that folded sessions do not have a negative impact on intrusion detection. Moreover, by using folded sessions the detection rate increases and the false positives only are 1% higher. When using folded sessions, we found a higher detection rate for variations of these same short attacks. The use of reduced sessions is very helpful when detecting attack variations.

References

1. Warrender, C., Forrest, S., Pearlmutter, B.: Detecting intrusions using system calls: Alternative data models. In: IEEE Symposium on security and Privacy, IEEE Computer Society Press (1999)
2. Yeung, D.Y., Ding, Y.: Host-based intrusion detection using dynamic and static behavioral models. Pattern Recognition **Vol. 36** (2003) pp. 229–243
3. Lippman, R.P., Cunningham, R.K., Fried, D.J., Graf, I., Kendall, K.R., Webster, S.E., Zissman, M.A.: Results of the DARPA 1998 offline intrusion detection evaluation. slides presented at RAID 1999 Conference (1999)

4. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Massachusets Institute of Technology, Cambridge, Massachusets 02142 (1999)
5. Qiao, Y., Xin, X., Bin, Y., Ge, S.: Anomaly intrusion detection method based on hmm. ELECTRONIC LETTERS **38** (2002) 663–664
6. Wagner, D., Soto, P.: Mimicry attacks on host based intrusion detection systems. In: Ninth ACM Conference on Computer and Communications Security, Washington, DC, USA, ACM (2002) 255–265