Insider Threat Detection using Stream Mining and Graph Mining

Pallabi Parveen, Jonathan Evans, Bhavani Thuraisingham,
Kevin W. Hamlen, and Latifur Khan
Department of Computer Science
The University of Texas at Dallas

***Abstract*—Evidence of malicious insider activity is often buried within large data streams, such as system logs accumulated over months or years. Ensemble-based stream mining leverages multiple classification models to achieve highly accurate anomaly detection in such streams even when the stream is unbounded, evolving, and unlabeled. This makes the approach effective for identifying insider threats who attempt to conceal their activities by varying their behaviors over time. This paper applies ensemble-based stream mining, unsupervised learning, and graph-based anomaly detection to the problem of insider threat detection, demonstrating that the ensemble-based approach is significantly more effective than traditional single-model methods.**

***Index Terms*—anomaly detection; graph-based detection; insider threat; ensemble**

I. Introduction

There is a growing consensus within the intelligence community that malicious insiders are perhaps the most potent threats to information assurance in many or most organizations [1]–[4]. One traditional approach to the *insider threat* detection problem is *supervised learning*, which builds data classification models from training data. Unfortunately, the training process for supervised learning methods tends to be time-consuming and expensive, and generally requires large amounts of well-balanced training data to be effective. In our experiments we observe that less than 3% of the data in realistic datasets for this problem are associated with insider threats (the minority class); over 97% of the data is associated with non-threats (the majority class). Hence, models trained from such imbalanced data are likely to perform poorly on test datasets.

An alternative approach is *unsupervised learning*, which can be effectively applied to purely unlabeled data—i.e., data in which no points are explicitly identified as anomalous or non-anomalous. *Graph-based anomaly detection (GBAD)* is one important form of unsupervised learning [5]–[7], but has traditionally been limited to static, finite-length datasets. This limits its application to streams related to insider threats, which tend to have unbounded length and threat patterns that evolve over time. Applying GBAD to the insider threat problem therefore requires a model that is sufficiently adaptive and efficient that effective models can be built from vast amounts of evolving data.

In this paper we cast insider threat detection as a stream mining problem and propose an efficient solution for applying unsupervised learning to detect anomalies in streams. Our approach combines multiple GBAD models in an *ensemble* of classifiers. The ensemble is designed to increase the classification accuracy relative to any single model over time. This evolutionary capability improves the classifier's survival of *concept-drift* as the behavior of both malicious and non-malicious agents varies over time. In experiments, we use test data that records system call data for a large, Unix-based, multiuser system.

Our work makes the following contributions: First, it shows how stream mining can be effectively applied to detect insider threats. Second, it proposes an unsupervised learning algorithm that copes with changes based on GBAD. Third, we exploit the power of stream mining and graph-based mining by effectively combining the two in a unified manner. This is the first work to our knowledge to harness these two approaches for insider threat detection. Finally, we compare our approach with traditional unsupervised learning approaches and show its su-

perior effectiveness.

The remainder of the paper is organized as follows. Section II presents related work. Section III presents our ensemble-based approach, and §IV discusses its relation to prior work in graph-based mining. Section V describes our experiments and testing methodology. Finally, §VI concludes with an assessment of the viability of ensemble-based mining for real-world insider threat detection.

II. Related Work

Insider threat detection work has applied ideas from both intrusion detection and external threat detection [8]–[11]. Supervised learning approaches collect system call trace logs containing records of normal and anomalous behavior [12]–[15], extract n-gram features from the collected data, and use the extracted features to train classifiers. Text classification approaches treat each system call as a word in a bag-of-words model [16]. Various attributes of system calls, including arguments, object path, return value, and error status, have been exploited as features in various supervised learning methods [17], [18]. These supervised approaches require large quantities of labeled training data and apply only to static, non-evolving streams.

Past work has also explored unsupervised learning for insider threat detection, but only to static streams to our knowledge [19]–[21]. Static GBAD approaches [5]–[7], [22] represent threat and non-threat data as a graph and apply unsupervised learning to detect anomalies. The *minimum description length (MDL)* approach to GBAD has been applied to email, cell phone traffic, business processes, and cybercrime datasets [23], [24]. Our work builds upon GBAD and MDL to support dynamic, evolving streams.

Stream mining [25] is a relatively new category of data mining research that applies to continuous data streams. In such settings, both supervised and unsupervised learning must be adaptive in order to cope with data whose characteristics change over time. There are two main approaches to adaptation: *incremental learning* [26] and *ensemble-based learning* [25], [27], [28]. Past work has demonstrated that ensemble-based approaches are the more effective of the two, motivating our approach.

TABLE I
Comparison of related work

Approach	(Un)Supervised	concept-drift	insider threat	graph-based
[12], [13]	S	×	✓	×
[20]	S	✓	×	×
[19], [21]	U	×	✓	×
GBAD [5], [6]	U	×	✓	✓
stream [25], [28], [29]	S	✓	N/A	N/A
ensemble (ours)	U	✓	✓	✓

Ensembles have been used in the past to bolster the effectiveness of positive/negative classification [28], [29]. By maintaining an ensemble of K models that collectively vote on the final classification, *false negatives (FN)* and *false positives (FP)* for a test set can be reduced. As better models are created, poorer models are discarded to maintain an ensemble of size exactly K. This helps the ensemble evolve with the changing characteristics of the stream and keeps the classification task tractable.

A comparison of the above related works is summarized in Table I. A more complete survey is available in [30].

III. Ensemble-based Insider Threat Detection

Data relevant to insider threats is typically accumulated over many years of organization and system operations, and is therefore best characterized as an unbounded data stream. Such a stream can be partitioned into a sequence of discrete *chunks*; for example, each chunk might comprise a week's worth of data.

Figure 1 illustrates how a classifier's decision boundary changes when such a stream observes concept-drift. Each circle in the picture denotes a data point, with unfilled circles representing *true negatives (TN)* (i.e., non-anomalies) and solid circles representing *true positives (TP)* (i.e., anomalies). The solid line in each chunk represents the decision boundary for that chunk, while the dashed line represents the decision boundary for the previous chunk.

Shaded circles are those that embody a new concept that has drifted relative to the previous chunk. In order to classify these properly, the decision boundary must be adjusted to account for the

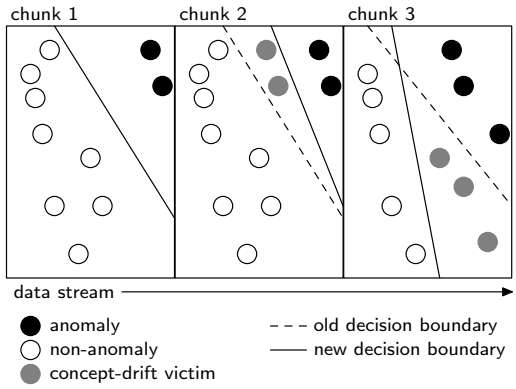

Fig. 1. Concept drift in stream data

new concept. There are two possible varieties of *misapprehension* (false detection):

1) The decision boundary of chunk 2 moves upward relative to chunk 1. As a result, some non-anomalous data is incorrectly classified as anomalous, causing the FP (false positive) rate to rise.
2) The decision boundary of chunk 3 moves downward relative to chunk 2. As a result, some anomalous data is incorrectly classified as non-anomalous, causing the FN (false negative) rate to rise.

In general, the old and new decision boundaries can intersect, causing both of the above cases to occur simultaneously for the same chunk. Therefore, both FP and FN counts may increase.

These observations suggest that a model built from a single chunk or any finite prefix of chunks is inadequate to properly classify all data in the stream. This motivates the adoption of our ensemble approach, which classifies data using an evolving set of K models.

The ensemble classification procedure is illustrated in Figure 2. We first use static, supervised GBAD to train models from an individual chunk. GBAD identifies *normative substructures* in the chunk, each represented as a subgraph. To identify an anomaly, a test substructure is compared against each model in the ensemble. Each model classifies the test substructure based on how much the test differs from the model's normative substructure. Once all models cast their votes, weighted majority voting is applied to make a final classification decision.

Ensemble evolution is arranged so as to maintain

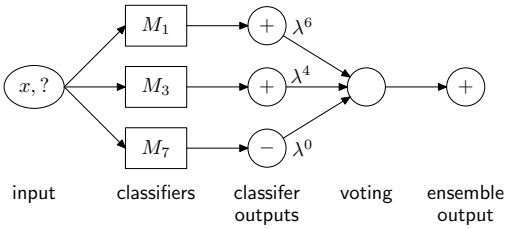

Fig. 2. Ensemble classification

a set of exactly K models at all times. As each new chunk arrives, a $K+1$st model is created from the new chunk and one victim model of these $K+1$ models is discarded. The discard victim can be selected in a number of ways. One approach is to calculate the prediction error of each of the $K+1$ models on the most recent chunk and discard the poorest predictor. This requires the *ground truth* to be immediately available for the most recent chunk so that prediction error can be accurately measured. If the ground truth is not available, we instead rely on majority voting; the model with least agreement with the majority decision is discarded. This results in an ensemble of the K models that best match the current concept.

Algorithm 1 summarizes the classification and ensemble updating algorithm. Lines 1–2 build a new model from the most recent chunk and temporarily add it to the ensemble. Next, Lines 3–9 apply each model in the ensemble to test graph t for possible anomalies. We use three varieties of GBAD for each model (P, MDL, and MPS), each discussed in §IV. Finally, Lines 10–18 update the ensemble by discarding the model with the most disagreements from the weighted majority opinion. If multiple models have the most disagreements, an arbitrary poorest-performing one is discarded.

Weighted majority opinions are computed in Line 11 using the formula

$$WA(E,a) = \frac{\sum_{\{i \,|\, M_i \in E,\, a \in A_{M_i}\}} \lambda^{\ell-i}}{\sum_{\{i \,|\, M_i \in E\}} \lambda^{\ell-i}} \qquad (1)$$

where $M_i \in E$ is a model in ensemble E that was trained from chunk i, A_{M_i} is the set of anomalies reported by model M_i, $\lambda \in [0,1]$ is a constant *fading factor* [31], and ℓ is the index of the most recent chunk. Model M_i's vote therefore receives weight $\lambda^{\ell-i}$, with the most recently constructed

Algorithm 1: Ensemble classification and updating

Input: E (ensemble)
t (test graph)
S (chunk)
Output: A (anomalies)
E' (updated ensemble)

1 $M' \leftarrow NewModel(S)$ // build new model
2 $E' \leftarrow E \cup \{M'\}$ // add model to ensemble
3 **foreach** $M \in E'$ **do** // for each model
4 $c_M \leftarrow 0$
5 **foreach** $q \in M$ **do** // find anomaly candidates
6 $A_1 \leftarrow GBAD_P(t, q)$
7 $A_2 \leftarrow GBAD_{MDL}(t, q)$
8 $A_3 \leftarrow GBAD_{MPS}(t, q)$
9 $A_M \leftarrow ParseResults(A_1, A_2, A_3)$
10 **foreach** $a \in \bigcup_{M \in E'} A_M$ **do** // for each candidate
11 **if** $round(WA(E', a)) = 1$ **then** // if anomaly
12 $A \leftarrow A \cup \{a\}$
13 **foreach** $M \in E'$ **do** // approbate yes-voters
14 **if** $a \in A_M$ **then** $c_M \leftarrow c_M + 1$
15 **else** // if non-anomaly
16 **foreach** $M \in E'$ **do** // approbate no-voters
17 **if** $a \notin A_M$ **then** $c_M \leftarrow c_M + 1$
18 $E' \leftarrow E' - \{choose(\arg\min_M(c_M))\}$ // drop worst model

model receiving weight $\lambda^0 = 1$, the model trained from the previous chunk receiving weight λ^1 (if it still exists in the ensemble), etc. This has the effect of weighting the votes of more current models above those of potentially outdated ones when $\lambda < 1$. Weighted average $WA(E, a)$ is then rounded to the nearest integer (0 or 1) in Line 11 to obtain the weighted majority vote.

For example, in Figure 2, models M_1, M_3, and M_7 vote positive, positive, and negative, respectively, for input sample x. If $\ell = 7$ is the most recent chunk, these votes are weighted λ^6, λ^4, and 1, respectively. The weighted average is therefore $WA(E, x) = (\lambda^6 + \lambda^4)/(\lambda^6 + \lambda^4 + 1)$. If $\lambda \leq 0.86$, the negative majority opinion wins in this case; however, if $\lambda \geq 0.87$, the newer model's vote outweighs the two older dissenting opinions, and the result is a positive classification. Parameter λ can thus be tuned to balance the importance of large amounts of older information against smaller amounts of newer information.

Our approach uses the results from previous iterations of GBAD iterations to identify anomalies in subsequent data chunks. That is, normative substructures found in previous GBAD iterations may persist in each model. This allows each model to consider all data since the model's introduction to the ensemble, not just that of the current chunk. When streams observe concept-drift, this can be a significant advantage because the ensemble can identify patterns that are normative over the entire data stream or a significant number of chunks but not in the current chunk. Thus, insiders whose malicious behavior is infrequent can still be detected.

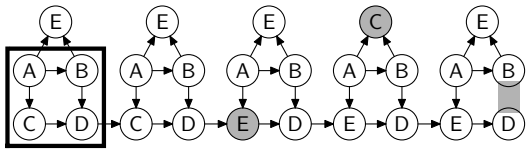

Fig. 3. A graph with a normative substructure (boxed) and anomalies (shaded)

It is important to note that the size of the ensemble remains fixed over time. Outdated models that are performing poorly are replaced by better-performing, newer models that are more suited to the current concept. This keeps each round of classification tractable even though the total amount of data in the stream is potentially unbounded.

IV. Graph-based Anomaly Detection

Algorithm 1 uses three varieties of GBAD to infer potential anomalies using each model. GBAD is a graph-based approach to finding anomalies in data by searching for three factors: modifications, insertions, and deletions of vertices and edges. Each unique factor runs its own algorithm that finds a normative substructure and attempts to find the substructures that are similar but not completely identical to the discovered normative substructure. A normative substructure is a recurring subgraph of vertices and edges that, when coalesced into a single vertex, most compresses the overall graph. The rectangle in Figure 3 identifies an example of normative substructure for the depicted graph.

Our implementation uses SUBDUE [32] to find normative substructures. The best normative substructure can be characterized as the one with minimal description length (MDL):

$$L(S, G) = DL(G \mid S) + DL(S) \qquad (2)$$

where G is the entire graph, S is the substructure being analyzed, $DL(G \mid S)$ is the description length of G after being compressed by S, and $DL(S)$ is

the description length of the substructure being analyzed. Description length $DL(G)$ is the minimum number of bits necessary to describe graph G [33].

Insider threats appear as small percentage differences from the normative substructures. This is because insider threats attempt to closely mimic legitimate system operations except for small variations embodied by illegitimate behavior. We apply three different approaches for identifying such anomalies, discussed below.

A. GBAD-MDL

Upon finding the best compressing normative substructure, GBAD-MDL searches for deviations from that normative substructure in subsequent substructures. By analyzing substructures of the same size as the normative one, differences in the edges and vertices' labels and in the direction or endpoints of edges are identified. The most anomalous of these are those substructures for which the fewest modifications are required to produce a substructure isomorphic to the normative one. In Figure 3, the shaded vertex labeled E is an anomaly discovered by GBAD-MDL.

B. GBAD-P

In contrast, GBAD-P searches for insertions that, if deleted, yield the normative substructure. Insertions made to a graph are viewed as extensions of the normative substructure. GBAD-P calculates the probability of each extension based on edge and vertex labels, and therefore exploits label information to discover anomalies. The probability is given by

$$P(A{=}v) = P(A{=}v \mid A)\, P(A) \tag{3}$$

where A represents an edge or vertex attribute and v represents its value. Probability $P(A{=}v \mid A)$ can be generated by a Gaussian distribution:

$$\rho(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{4}$$

where μ is the mean and σ is the standard deviation. Higher values of $\rho(x)$ correspond to more anomalous substructures.

Using GBAD-P therefore ensures that malicious insider behavior that is reflected by the actual data in the graph (rather than merely its structure) can be reliably identified as anomalous by our algorithm. In Figure 3, the shaded vertex labeled C is an anomaly discovered by GBAD-P.

```
header,150,2,execve(2),,Fri Jul 31 07:46:33 1998, +
        562468777 msec
path/usr/lib/fs/ufs/quota
attribute,104555,root,bin,8388614,187986,0
exec_args,1,
/usr/sbin/quota
subject,2110,root,rjm,2110,rjm,280,272, +
        0 0 172.16.112.50
return,success,0
trailer,150
```

Fig. 4. A sample system call record from the MIT Lincoln dataset

C. GBAD-MPS

Finally, GBAD-MPS considers deletions that, if re-inserted, yield the normative substructure. To discover these, GBAD-MPS examines the parent structure. Changes in size and orientation in the parent signify deletions amongst the subgraphs. The most anomalous substructures are those with the smallest transformation cost required to make the parent substructures identical. In Figure 3, the last substructure of A-B-C-D vertices is identified as anomalous by GBAD-MPS because of the missing edge between B and D marked by the shaded rectangle.

V. Experiments

We tested our algorithm on the 1998 Lincoln Laboratory Intrusion Detection dataset [34]. This dataset consists of daily system logs containing all system calls performed by all processes. It was created using the Basic Security Mode (BSM) auditing program. Each log consists of tokens that represent system calls using the syntax exemplified in Figure 4.

The token arguments begin with a header line and end with a trailer line. The header line reports the size of the token in bytes, a version number, the system call, and the date and time of execution in milliseconds. The second line reports the full path name of the executing process. The optional `attribute` line identifies the user and group of the owner, the file system and node, and the device. The next line reports the number of arguments to the system call, followed by the arguments themselves on the following line. The `subject` line reports the audit ID, effective user and group IDs, real user and

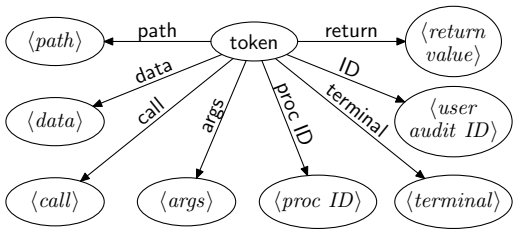

Fig. 5. A token subgraph

group IDs, process ID, session ID, and terminal port and address, respectively. Finally, the `return` line reports the outcome and return value of the system call.

Since many system calls are the result of automatic processes not associated with any particular user, for insider threat detection we limit our attention to user-affiliated system calls. These include calls for `exec`, `execve`, `utime`, `login`, `logout`, `su`, `rsh`, `rexecd`, `passwd`, `rexd`, and `ftp`. All of these correspond to logging in/out or file operations performed by users, and are therefore relevant to insider threat detection. Restricting our attention to such operations helps to reduce extraneous noise in the dataset. The subgraph yielded by extracting attributes from these tokens is depicted in Figure 5.

Each of the attributes in Figure 5 are important for different reasons. For example, the path value may indicate the importance or security level of the information being accessed. A change in the file path access locations or the type of system calls being executed by a given user might indicate anomalous behavior that should be investigated. File paths for `exec` and `execve` calls reveal the various programs executed by each user. Process IDs allow the tracking of all system calls made by any given process, indicating the behavior of that process over time. Terminal information provides a form of locality data for users. If a user logs in from an unexpected location or displays anomalous behavior from certain locations, this could indicate the use of stolen credentials or other malicious activity.

Table II reports statistics for the dataset after all irrelevant tokens have been filtered out and the attribute data in Figure 5 has been extracted. Preprocessing extracted 62K tokens spanning 500K vertices. These reflected the activity of all users over 9 weeks. We applied our ensemble learning algorithm to this data with the number of models (K) set to 1, 3, 5, 7, and 9 respectively. During each ensemble iteration, each model chose the best 5 normative substructures to use for anomaly detection. It is possible to configure the algorithm to use more than 5, but this default yielded good results so we left it unchanged.

TABLE II
DATASET STATISTICS AFTER FILTERING AND ATTRIBUTE EXTRACTION

Statistic	Value
# vertices	500,000
# tokens	62,000
# normative substructures	5
# users	all
duration	9 weeks

TABLE III
TOTAL ENSEMBLE ACCURACY

models	TP	FP	FN
$K = 1$	9	920	0
$K = 3$	9	188	0
$K = 5$	9	180	0
$K = 7$	9	179	0
$K = 9$	9	150	0

Performance was measured in terms of total false positives (FP) and false negatives (FN). We chose the Lincoln dataset both because of its large size and because its set of anomalies is well known, facilitating an accurate performance assessment via misapprehension counts. Table III summarizes the results, along with a count of true positives (TP). The line for $K = 1$ reflects the performance of a non-ensemble, single-model GBAD approach. Using ensemble learning with $K = 9$ models yielded 84% fewer false positives, significantly reducing the false alarm rate. In all cases there were no false negatives; all anomalies were detected.

The greatest impact of increasing the model count was observed from $K = 1$ to $K = 3$. This shows that ensemble learning provides significant improvements over single-model approaches even when the ensemble size is small. It also reveals that when computation time is at a premium, small ensembles can potentially provide high classification accuracy with reasonable runtimes. In our experiments, using

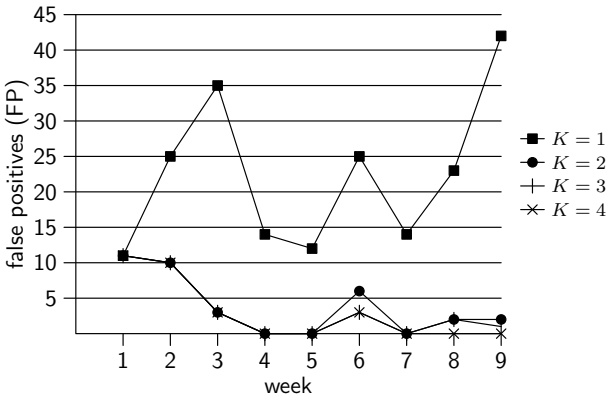

Fig. 6. Distribution of false positives

$K = 3$ models achieved almost the same reduction in false positives (an 80% reduction) as the $K = 9$ case, but with less than a 10th of the running time.

Figure 6 shows the distribution of false positives raised by each ensemble over time. In this dataset, all true insider threat activity occurs during week 6, leading to the spike in false positives for that week. During the first 5 weeks, the ensemble approaches progressively improve as they learn legitimate patterns of behavior, whereas the single-model approach struggles, oscillating wildly. When the malicious insider behavior appears in week 6, the false positive rate rises for both approaches as they attempt to learn the new concept. The ensemble approaches largely succeed, yielding low false positive rates for the ensuing weeks, but the single-model approach has become corrupted. It cannot adequately encode concepts related to both legitimate and illegitimate behavior in the same model, leading to the steep increase in false positives by week 9.

We next investigate the impact of parameters K (the ensemble size) and q (the number of normative substructures per model) on the classification accuracy and running times. To more easily perform the larger number of experiments necessary to chart these relationships, we employ the smaller datasets summarized in Table IV for these experiments. Dataset A consists of activity associated with user `donaldh` during weeks 2–8, and dataset B is for user `william` during weeks 4–7. Each of these users displays malicious insider activity during the respective time periods. Both datasets evince similar trends for all relationships discussed henceforth; therefore we report only the details for dataset A throughout the remainder of the section.

TABLE IV
SUMMARY OF DATASETS A AND B

Statistic	**Dataset** A	**Dataset** B
user	donaldh	william
# vertices	269	1283
# edges	556	469
week	2–8	4–7
weekday	Friday	Thursday

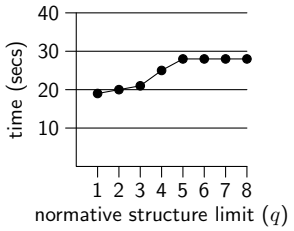

Fig. 7. The effect of q on runtimes for fixed $K = 6$ on dataset A

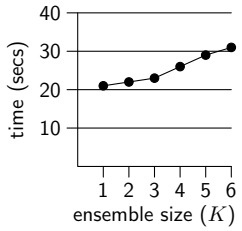

Fig. 8. The effect of K on runtimes for fixed $q = 4$ on dataset A

Figure 7 shows the relationship between the cutoff q for the number of normative substructures and the running time in dataset A. Times increase approximately linearly until $q = 5$ because there are only 4 normative structures in dataset A. The search for a 5th structure therefore fails (but contributes running time), and higher values of q have no further effect.

Figure 8 shows the impact of ensemble size K and runtimes for dataset A. As expected, runtimes increase approximately linearly with the number of models (2 seconds per model on average in this dataset).

Increasing q and K also tends to aid in the discovery of true positives (TP). Figures 9 and 10 illustrate by showing the positive relationships of q and K, respectively, to TP. Once $q = 4$ normative substructures are considered per model and $K = 4$ models are consulted per ensemble, the classifier reliably detects all 7 true positives in dataset A. These values of q and K therefore strike the best balance between coverage of all insider threats and the efficient runtimes necessary for high responsiveness.

Increasing q to 4 does come at the price of raising more false alarms, however. Figure 11 shows that the false positive rate increases along with the true

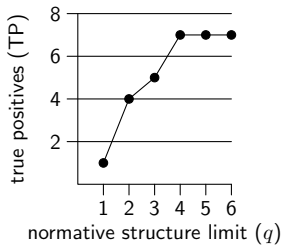

Fig. 9. The effect of q on TP rates for fixed $K = 6$ on dataset A

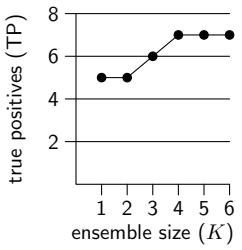

Fig. 10. The effect of K on TP rates for fixed $q = 4$ on dataset A

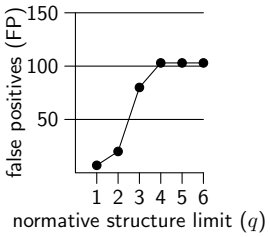

Fig. 11. The effect of q on FP rates for fixed $K = 6$ on dataset A

positive rate until $q = 4$. Dataset A has only 4 normative structures, so increasing q beyond this point has no effect.

Table V considers the impact of weighted versus unweighted majority voting on the classification accuracy for fixed $q = 3$. The unweighted columns are those for $\lambda = 1$, and the weighted columns use fading factor $\lambda = 0.9$. The dataset consists of all tokens associated with user `donaldh`. Weighted majority voting has no effect in these experiments except when $K = 4$, where it reduces the FP rate from 124 (unweighted) to 85 (weighted) and increases the TN rate from 51 (unweighted) to 90 (weighted). However, since these results can be obtained for $K = 3$ without weighted voting, we conclude that weighted voting merely serves to mitigate a poor choice of K; weighted voting has little or no impact when K is chosen wisely.

TABLE V
IMPACT OF FADING FACTOR λ (WEIGHTED VOTING)

	$K = 2$		$K = 3$		$K = 4$	
	λ=1	λ=0.9	λ=1	λ=0.9	λ=1	λ=0.9
TP	10	10	10	10	14	14
FP	79	79	85	85	124	85
TN	96	96	90	90	51	90
FN	4	4	4	4	0	0

VI. Conclusion and Future Work

The ensemble-based approach to insider threat detection succeeded in identifying all anomalies in the 1998 Lincoln Laboratory Intrusion Detection dataset with zero false negatives and a lower false positive rate than related single-model approaches. The technique combines the non-supervision advantages of graph-based anomaly detection with the adaptiveness of stream mining to achieve effective, practical insider threat detection for unbounded, evolving data streams.

Future work should consider a wider range of tunable parameters in an effort to further reduce false positive rates, for which there is still substantial room for improvement. In addition, a more sophisticated polling algorithm that weights the votes of better models might have significant accuracy advantages.

Reducing classifier runtimes is also important for rapidly detecting and responding to emerging insider threats. The experiments in this paper were conducted with a purely serial implementation, but the ensemble algorithm includes substantial room for parallelization. Future work should therefore investigate the application of distributed and cloud computing technologies to improve efficiency.

Finally, although practical insider threat detection mechanisms must realistically assume that ground truth in the form of labeled data points is generally unavailable, it would be beneficial to take advantage of such labels if they become available over time. Future research should therefore examine techniques for updating the models within an ensemble with supervised learning to improve the classification accuracy for partially labeled streams.

Acknowledgment

This material is based upon work supported by The Air Force Office of Scientific Research under Award No. FA-9550-09-1-0468. We thank Dr. Robert Herklotz for his support.

References

[1] R. C. Brackney and R. H. Anderson, Eds., *Understanding the Insider Threat*. RAND Corporation, March 2004.

[2] M. P. Hampton and M. Levi, "Fast spinning into oblivion? recent developments in money-laundering policies and offshore finance centres," *Third World Quarterly*, vol. 20, no. 3, pp. 645–656, 1999.

[3] S. Matzner and T. Hetherington, "Detecting early indications of a malicious insider," *IA Newsletter*, vol. 7, no. 2, pp. 42–45, 2004.

[4] M. B. Salem and S. J. Stolfo, "Modeling user search behavior for masquerade detection," in *Proc. Recent Advances in Intrusion Detection (RAID)*, 2011, forthcoming.

[5] D. J. Cook and L. B. Holder, Eds., *Mining Graph Data*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2007.

[6] W. Eberle and L. B. Holder, "Mining for structural anomalies in graph-based data," in *Proc. International Conference on Data Mining (DMIN)*, 2007, pp. 376–389.

[7] D. J. Cook and L. B. Holder, "Graph-based data mining," *IEEE Intelligent Systems*, vol. 15, no. 2, pp. 32–41, 2000.

[8] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus, and Y. Vardi, "Computer intrusion: Detecting masquerades," *Statistical Science*, vol. 16, no. 1, pp. 1–17, 2001.

[9] K. Wang and S. J. Stolfo, "One-class training for masquerade detection," in *Proc. ICDM Workshop on Data Mining for Computer Security (DMSEC)*, 2003.

[10] R. A. Maxion, "Masquerade detection using enriched command lines," in *Proc. IEEE International Conference on Dependable Systems & Networks (DSN)*, 2003, pp. 5–14.

[11] E. E. Schultz, "A framework for understanding and predicting insider attacks," *Computers and Security*, vol. 21, no. 6, pp. 526–531, 2002.

[12] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for Unix processes," in *Proc. IEEE Symposium on Computer Security and Privacy (S&P)*, 1996, pp. 120–128.

[13] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol. 6, no. 3, pp. 151–180, 1998.

[14] N. Nguyen, P. Reiher, and G. H. Kuenning, "Detecting insider threats by monitoring system call activity," in *Proc. IEEE Information Assurance Workshop (IAW)*, 2003, pp. 45–52.

[15] D. Gao, M. K. Reiter, and D. Song, "On gray-box program tracking for anomaly detection," in *Proc. USENIX Security Symposium*, 2004, pp. 103–118.

[16] Y. Liao and V. R. Vemuri, "Using text categorization techniques for intrusion detection," in *Proc. 11th USENIX Security Symposium*, 2002, pp. 51–59.

[17] C. Krügel, D. Mutz, F. Valeur, and G. Vigna, "On the detection of anomalous system call arguments," in *Proc. 8th European Symposium on Research in Computer Security (ESORICS)*, 2003, pp. 326–343.

[18] G. Tandon and P. Chan, "Learning rules from system call arguments and sequences for anomaly detection," in *Proc. ICDM Workshop on Data Mining for Computer Security (DMSEC)*, 2003, pp. 20–29.

[19] A. Liu, C. Martin, T. Hetherington, and S. Matzner, "A comparison of system call feature representations for insider threat detection," in *Proc. IEEE Information Assurance Workshop (IAW)*, 2005, pp. 340–347.

[20] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data," in *Applications of Data Mining in Computer Security*, D. Barbará and S. Jajodia, Eds. Springer, 2002, ch. 4.

[21] E. Eskin, M. Miller, Z.-D. Zhong, G. Yi, W.-A. Lee, and S. Stolfo, "Adaptive model generation for intrusion detection systems," in *Proc. ACM CCS Workshop on Intrusion Detection and Prevention (WIDP)*, 2000.

[22] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *Proc. International Conference on Data Mining (ICDM)*, 2002, pp. 721–724.

[23] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "GrIDS—a graph based intrusion detection system for large networks," in *Proc. 19th National Information Systems Security Conference*, 1996, pp. 361–370.

[24] E. Kowalski, T. Conway, S. Keverline, M. Williams, D. Cappelli, B. Willke, and A. Moore, "Insider threat study: Illicit cyber activity in the government sector," U.S. Department of Homeland Security, U.S. Secret Service, CERT, and the Software Engineering Institute (Carnegie Mellon University), Tech. Rep., January 2008.

[25] W. Fan, "Systematic data selection to mine concept-drifting data streams," in *Proc. ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2004, pp. 128–137.

[26] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2000, pp. 71–80.

[27] M. M. Masud, Q. Chen, J. Gao, L. Khan, C. Aggarwal, J. Han, and B. Thuraisingham, "Addressing concept-evolution in concept-drifting data streams," in *Proc. IEEE International Conference on Data Mining (ICDM)*, 2010, pp. 929–934.

[28] M. M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham, "Classification and novel class detection in concept-drifting data streams under time constraints," *IEEE Trans. Knowl. Data Eng. (TKDE)*, vol. 23, no. 6, pp. 859–874, 2011.

[29] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "A practical approach to classify evolving data streams: Training with limited amount of labeled data," in *Proc. IEEE International Conference on Data Mining (ICDM)*, 2008, pp. 929–934.

[30] M. B. Salem, S. Herkshkop, and S. J. Stolfo, "A survey of insider attack detection research," *Insider Attack and Cyber Security*, vol. 39, pp. 69–90, 2008.

[31] L. Chen, S. Zhang, and L. Tu, "An algorithm for mining frequent items on data stream using fading factor," in *Proc. IEEE International Computer Software and Applications Conference (COMPSAC)*, 2009, pp. 172–177.

[32] N. S. Ketkar, L. B. Holder, and D. J. Cook, "Subdue: Compression-based frequent pattern discovery in graph data," in *Proc. ACM KDD Workshop on Open-Source Data Mining*, 2005.

[33] W. Eberle, J. Graves, and L. Holder, "Insider threat detection using a graph-based approach," *Journal of Applied Security Research*, vol. 6, no. 1, pp. 32–81, 2011.

[34] K. Kendall, "A database of computer attacks for the evaluation of intrusion detection systems," Master's thesis, Massachusetts Institute of Technology, 1998.