Supervised Learning for Insider Threat Detection Using Stream Mining

Pallabi Parveen, Zackary R Weger, Bhavani Thuraisingham, Kevin Hamlen and Latifur Khan
Department of Computer Science
University of Texas at Dallas

***Abstract*—Insider threat detection requires the identification of rare anomalies in contexts where evolving behaviors tend to mask such anomalies. This paper proposes and tests an ensemble-based stream mining algorithm based on supervised learning that addresses this challenge by maintaining an evolving collection of multiple models to classify dynamic data streams of unbounded length. The result is a classifier that exhibits substantially increased classification accuracy for real insider threat streams relative to traditional supervised learning (traditional SVM and one-class SVM) and other single-model approaches.**

Keywords- anomaly detection, support vector machine, insider threat, ensemble

I. INTRODUCTION

Insider threats are increasingly cited as among the most potent dangers to modern computing infrastructures [40]. Reliable detection of insider threats is particularly challenging because insiders mask and adapt their behaviors to resemble legitimate system and organizational activities. One approach to detecting these threats is supervised learning, which builds models from training data. However, supervised learning requires a potentially expensive training process [41], and is therefore impeded by the typically small quantity of insider threat data available for such training. For example, in our insider threat dataset, only about 0.03% of the training data is associated with insider threats (the minority class), while 99.97% of the data is associated with non-threats (the majority class). Traditional support vector machines (SVM) [33, 42, 43] trained from such an imbalanced dataset are likely to perform poorly on test datasets.

One-class SVMs (OCSVM) [42, 43] address the rare-class issue by building a model that considers only normal data (i.e., non-threat data). During the testing phase, test data is classified as normal or anomalous based on geometric deviations from the model. However, the approach is only applicable to bounded-length, static data streams. In contrast, insider threat-related data is typically continuous, and threat patterns evolve over time. In other words, the data is a stream of unbounded length. Hence, effective classification models must be adaptive (i.e., able to cope with evolving concepts) and highly efficient in order to build the model from large amounts of evolving data.

We therefore conceptualize the insider threat detection problem as a stream mining problem, and we approach this problem through the efficient detection of anomalies in stream data. To cope with concept-evolution, our approach maintains an evolving ensemble of multiple OCSVM models. The ensemble updating process keeps the ensemble current as the stream evolves, preserving high classification accuracy as both legitimate and illegitimate behaviors evolve over time. Test data consisting of real-time recorded data of raw system calls is used to demonstrate the practicality of the approach.

Our primary contributions are as follows: First, we show how stream mining can be effectively used to detect insider threats. Second, we propose a supervised learning solution that copes with evolving concepts using one-class SVMs. Third, we effectively address the challenge of limited labeled training data (rare instance issues). Finally, we compare our approach with traditional supervised learning approaches and show the effectiveness of our approach using real-world insider threat data.

Section II begins with a presentation of related work, Section III then presents our proposed approach *Ensemble based Insider Threat (EIT)* and Section IV discusses the background of OCSVM. Section V describes our experiments and approaches to testing our ensemble. Section VI presents our performance results. Finally, Section VII concludes and suggests future work.

II. RELATED WORK

Insider threat detection work has utilized ideas from intrusion detection or external threat detection areas [8, 9, 10]. For example, supervised learning has been applied to detect insider threats. System call traces from normal activity and anomaly data are gathered [13, 14]; features are extracted from this data using n-gram and finally, trained with classifiers. Lia and Vemuri exploit text classification idea in insider threat domain [15] where each system call is treated as a word in bag of words model. System call, and related attributes, arguments, object path, return value and error status of each system call are served as features in various supervised methods [17, 18]. A supervised model based on hybrid high-order Markov chain model was adopted by Ju and Vardi [35]. First, a *signature behavior* for a particular user based on the command sequences that the user executed is identified and then anomaly is detected.

Stolfo et al. [32] introduced a general purpose algorithm for anomaly detection (in windows environment), the Probabilistic Anomaly Detection (PAD) algorithm, that assumes anomalies or noise are a rare event in the training data.

Schonlau et al. applied a number of detection methods to a data set of "truncated" UNIX shell commands for 70 users [37]. Commands were collected using the UNIX acct auditing

mechanism. For each user a number of commands were gathered over a period of time (http://www.schonlau.net). The detection methods are supervised based on multistep markovian model, and combination of Bayes and Markov approach. Maxion [36] argued that Schonlau data set was not appropriate for the masquerade detection task. First, the period of data gathering for a user varied from one user to the other a lot (from several days to several months). Furthermore, commands are not logged in the order in which they are typed, but rather when the application ends the audit mechanism used to collect the data. Hence, during the feature extraction this method focuses on strict sequence analysis that may be faulty. Therefore, in this proposed work we have not considered this dataset.

These approaches differ from our work in the following way. These learning approaches are static in nature and do not learn over evolving stream. In other words, stream characteristics of data are not explored further. Hence, static learner performance may degrade over time. On the other hand, our approach will learn from evolving data stream.

Researchers have explored *unsupervised learning* [5] for insider threat detection. However, this learning algorithm is static in nature. On the other hand, our approach is supervised and at the same time learns from evolving stream over time.

Researchers treat insider data and non insider data as a graph and apply unsupervised learning to detect anomaly. Here, much of the work with Graph Based Anomaly Detection (GBAD) was performed by Lawrence Holder of Washington State and his colleagues at Tennessee Tech [21, 23, 25]. The minimum description length (MDL) approach has been applied to email, cell-phone traffic, business processes and cybercrime datasets [24, 26]. It is not clear from their work how this GBAD framework can be extended in dynamic/evolving stream (dynamic graphs). In other words, the framework is static in nature. Our proposed work is based on supervised learning and it can handle dynamic data or stream data well by learning from evolving stream.

In anomaly detection one class SVM algorithm (OCSVM) is used [32]. OCSVM builds a model from training on normal data and then classifies a test data as benign or anomaly based on geometric deviations from normal training data. Wang and Stolfo et al. [32] showed, for masquerade detection, one-class SVM training is as effective as two-class training. The authors have investigated SVMs using binary features and frequency based features. The one-class SVM algorithm with binary features performed the best.

To find frequent patterns, Szymanski and Zhang [38] proposed recursive mining, encoded the patterns with unique symbols, and rewrote the sequence using this new coding. They used a one-class SVM classifier for masquerade detection.

To the best of our knowledge there is no work that extends this OCSVM in a stream domain. On the other hand, although our approach relies on OCSVM but it is extended to stream domain so that it can cope with changes.

Stream mining is a new data mining area where data is continuous [29]. In addition, characteristics of data may change over time (concept drift). Here, supervised and unsupervised learning need to be adaptive to cope with changes [27, 28, 29, 30]. There are two ways adaptive learning can be developed. One is incremental learning [30] and the other one is ensemble-based learning [27, 28, 29]. Here is an example for incremental learning in user action prediction. Davidson and Hirsch [34] introduced Incremental Probabilistic Action Modeling (IPAM), based on one-step command transition probabilities estimated from the training data. The probabilities were continuously updated with the arrival of a new command and modified with the usage of an exponential decay scheme. Therefore, to the best of our knowledge, there is no work from other researchers that handles insider threat detection in stream mining area. This is the first attempt to detect insider threat using ensemble based stream mining. In the literature, it is shown that ensemble-based learning is more effective than incremental learning [29]. Therefore, in this paper, we focus on ensemble-based technique instead of incremental learning.

Ensembles have been used previously to bolster the effectiveness of positive/negative classification [4, 28]. By maintaining an ensemble of K models that all have a say in the final outcome, false negatives and false positives for a test bed can be reduced. For every $K+1$ model that is created, one model is discarded to maintain only K of them. Instead of a single model deciding whether a result is correct, most or all of K models must do so. This reduces the chances of a decision being made too hastily.

In our previous work we have applied unsupervised learning to detect insider threat in a data stream [42]. The difference between this work and the former work is supervised learning and unsupervised learning. In result section we compare two learning approaches and show superiority of supervised learning in terms of quality of results.

We summarize all related work in Table 1.

III. ENSEMBLE BASED INSIDER THREAT (EIT)

Insider threat detection related data is stream in nature. For example, data may be gathered over years. Here, we assume continuous data stream will be converted into a number of chunks. For example, each chunk may represent a week and contain the data which arrived during that time period.

Figure 1 demonstrates how the classifier decision boundary is changing over time (from one chunk to next chunk). Data points are associated with two classes (normal and anomaly). There are three contiguous data chunks as shown in the Figure 1. The dark straight line represents the decision boundary of its own chunk, whereas the dotted straight line represents the decision boundary of previous chunk. If there were no concept drift in the data stream, the decision boundary should be the same for the current chunk and its previous chunk (the dotted straight line). White dots represent the normal data (True Negative), blue dots represent anomaly data (True Positive), and striped dots represent the instances victim of concept drift.

We show two different cases in the Figure 1 are as follows.

Case1: the decision boundary of the second chunk moves upwards compared to that of the first chunk. As a result, more normal data will be classified as anomaly by the decision boundary of the first chunk, thus FP will go up. Recall that a test point having true benign (normal) category is classified as an anomaly by a classifier is known as a FP.

Case2: the decision boundary of the third chunk moves downwards compared to that of the first chunk. So, more anomaly data will be classified as normal data by the decision boundary of the first chunk, thus FN will go up. Recall that a test point having true malicious category is classified as benign by a classifier is known as a FN.

In more general case, the decision boundary of the current chunk can vary which causes the decision boundary of the previous chunk to misclassify both normal data and anomaly data. Therefore, both FP and FN may go up at the same time.

Therefore, it suggests that a model build from a single chunk could not be sufficed. So, for detecting anomalies, we will exploit ensemble based model (here K models), namely EIT. From a chunk we will build a model using OCSVM (supervised manner). To identify an anomaly, a test point will be compared against each model of the ensemble. Recall that a model will declare the test data as an anomaly based on how much the test differs from the model's normative patterns. Once all models cast their votes, we will apply majority voting to make the final decision whether the test is an anomaly (as shown in Figure 2).

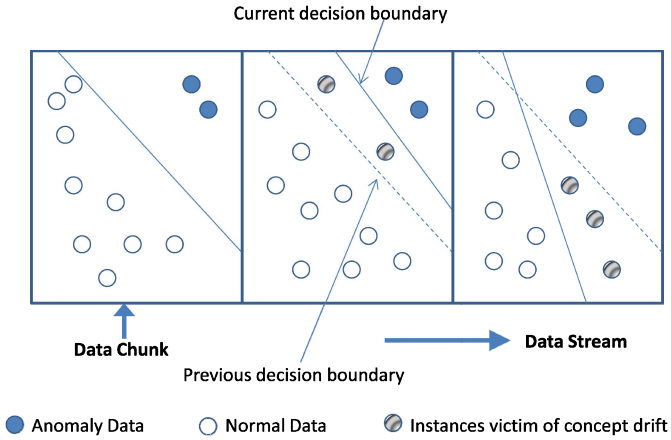

Figure 1. Concept drift in Stream Data

Model Update: We always keep an ensemble of fixed size models (K in that case). Hence, when a new chunk will be processed (see Figure 2), we already have K models in the ensemble and the new model will be created from the current chunk. We need to update ensemble by replacing a victim model with this new model. Victim selection can be done in a number of ways. One approach can be to calculate the prediction error of each model on the most recent chunk relative to the majority vote. Here, we assume ***ground truth*** on the most recent chunk is not available. If ground truth is available we can exploit that knowledge for training. The new model will replace an existing model from the ensemble which gives the maximum prediction error.

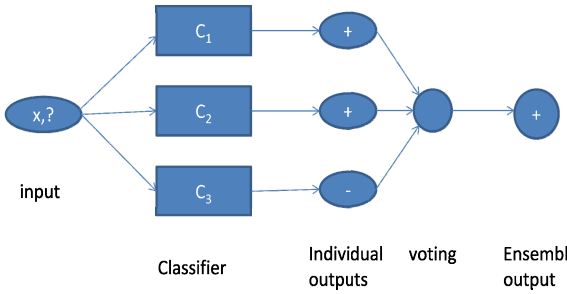

Figure 2. Architecture of an Ensemble Classifier Sketch

Thus, we keep track of an ensemble of K models.

Table 1. Approaches for Insider Threat Detection

Proposed techniques	Challenges		
	Supervised/ Unsupervised	Concept-drift	Insider Threat
Forrest[13], Hofmeyr [14], [35], [37]	Supervised	X	√
Liu [5]	Unsupervised	X	√
Holder (GBAD) [21, 23]	Unsupervised	X	√
One Class SVM [32]	Supervised	X	√
Masud [4, 28], Fan[29] (Stream Mining)	N/A	√	X
Our Previous Approach [42]	Unsupervised	√	√
Our Approach (EIT)	Supervised	√	√

Algorithm 1 shows basic building blocks of EIT algorithm. Here, first we present how we update the model. Input for the algorithm 1 will be as follows: D_n most recently labeled data chunk (most recent training chunk) and A the ensemble. From line 1 to 3 we calculate prediction error of each model on D_n.

Line 4 a new model will be built using OCSVM on D_n. After line 5 we have $(K+1)$ models and we need to add the new model and discard existing one based on maximum prediction error.

Algorithm 1 Updating the classifier ensemble
Input: D_n: the most recently labeled data chunks,
A: the current ensemble of best K classifiers
Output: an updated ensemble A

1: **for** each model $M \in A$ **do**
2: Test M on D_n and compute its expected error
3: **end for**
4: $M_n \leftarrow$ Newly trained 1-class SVM classifier (OCSVM) from data D_n
5: Test M_n on D_n and compute its expected error
6: $A \leftarrow$ best K classifiers from $M_n \cup A$ based on expected error

In Algorithm 2, we focus on testing using ensemble. Ensemble A and latest chunk on unlabeled instances D_u will be the input. At line 1 feature extraction and selection will be done from latest chunk of unlabeled instances. Between line 2 and line 6 we will take each data point from D_u and do the prediction. For this for a particular test point between lines 4 and 5, each model of the ensemble will be used to predict whether test data is an anomaly or not. Finally, at line 6 we predict anomalies based on majority voting.

Algorithm 2 Testing Algorithm
Input: $A \leftarrow$ Build-initial-ensemble()
$D_u \leftarrow$ latest chunk of unlabeled instances
Output: Prediction/Label of Du
1: $Fu \leftarrow$ **Extract&Select-Features**(D_u)
//Feature set for D_u
2: **for each** $x_j \in F_u$ **do**
3. Results$\leftarrow$NULL
4. **for** each model M in A
5. Results$\leftarrow$ Results U Prediction (x_j, M)
end for
6. Anomalies $\leftarrow$Majority Voting (Results)
end for

The EIT uses the results from previous iterations of OCSVM executions to identify anomalies in subsequent data chunks. This allows the consideration of more than just the current data being analyzed. Models found in previous OCSVM iterations are also analyzed, not just the models of the current dataset chunk. The ensemble handles the execution in this manner because patterns identified in previous chunks may be normative over the entire data stream or a significant number of chunks but not in the current execution chunk. Thus insiders whose malicious behavior is infrequent will be detected. It is important to note that we always keep our ensemble size fixed. Hence, an outdated model which is performing worst on the most recent chunks will be replaced by the new one.

IV. ONE CLASS SVM

In a chunk, a model is built using OCSVM [42, 43]. The OCSVM approach first maps training data into a high dimensional feature space (via a kernel). Next, the algorithm iteratively finds the maximal margin hyperplane which best separates the training data from the origin. The OCSVM may be considered as a regular two-class SVM. Here the first class entails all the training data, and the second class is the origin. Thus, the hyperplane (or linear decision boundary) corresponds to the classification rule:

$$f(x) = \langle w, x \rangle + b$$

where w is the normal vector and b is a bias term. The OCSVM solves an optimization problem to find the rule with maximal geometric margin. This classification rule will be used to assign a label to a test example x. If $f(x) < 0$, we label x as an anomaly, otherwise it is labeled normal. In reality there is a trade-off between maximizing the distance of the hyperplane from the origin and the number of training data points contained in the region separated from the origin by the hyperplane.

I. EXPERIMENTS

We tested EIT on the 1998 Lincoln Laboratory Intrusion Detection dataset [2]. This dataset contains daily logs of all system calls made over a 7 week period. It was created using Basic Security Mode auditing program (BSM). Each log consists of tokens for every system call made. The tokens can be displayed in plaintext using the praudit command.

```
header,129,2,execve(2),,Tue Jun 16 08:14:29 1998, +
518925003 msec
path,/opt/local/bin/tcsh
attribute,100755,root,other,8388613,79914,0
exec_args,1,
-tcsh
subject,2142,2142,rjm,2142,rjm,401,400,24
1 135.13.216.191
return,success,0
trailer,129
```

Figure 4. Sample praudit output data

Figure 4 shows a sample token generated with the praudit command. Each token begins with a header line and ends with a trailer line. The header line consists of "header", the size of the token in bytes, audit record version number, the system call, and the time and date of the system call. The path line, if it is available, consists of "path" and the full system path of execution. The attribute line, if it is available, consists of "attribute", the file access mode and type, owner user ID, owner group ID, file system ID, node ID, and device ID. The exec_args line, if it is available contains "exec_args" and the number of arguments. The line following the exec_args line contains the number of arguments specified in the previous line. The subject line consists of "subject", audit ID, effective

user ID, effective group ID, real user ID, real group ID, process ID, audit session ID, and the terminal port and IP address. The return line consists of "return", the error status of the system call, and the return value of the system call. The trailer line consists of "trailer" and the size of the token in bytes.

From this dataset we filter out calls specifically made by users. Most of the tokens are system calls that are not initiated by users and are therefore not pertinent to the detection of insider threats. Leaving these calls in the dataset would unnecessarily increase noise and processing time. We specifically pull out tokens with system calls to exec, execve, utime, login, logout, su, setegid, seteuid, setuid, rsh, rexecd, passwd, rexd, and ftp, as all of these are calls initiated by users. Further, we pull out calls that contain the user ID of one of the users on the inside (in this dataset, there are 6 such users). Some tokens contain calls made by users from the outside, via web servers, and are not pertinent to the detection of *insider* threats. Leaving these tokens in the dataset would unnecessarily increase noise and processing time

.

Time, userID, machine IP, command, argument, path, return
1 1:29669 6:1 8:1 21:1 32:1 36:0

Figure 5. Feature set extracted from praudit data in Figure 4.

Figure 5 shows the features extracted from the praudit output data in Figure 4. The first number is the classification of the token as either anomalous (-1) or normal (1). The classification is used by 2-class SVM for training the model, but is unused (although required) for one class SVM. The rest of the line is a list of index-value pairs, which are separated by a colon (":"). The index represent the dimension for use by SVM, and the value is the value of the token along that dimension. The value must be numeric. The list must be ascending by index. Indices that are missing are assumed to have a value of 0. Attributes which are categorical in nature (and can take the value of any one of N categories) are represented by N dimensions. In Figure 5, "1:29669" means that the time of day (in seconds) is 29669. "6:1" means that the user's ID (which is categorical) is 2142, "8:1" means that the machine IP address (also categorical) is 135.13.216.191, "21:1" means that the command (categorical) is execve, "32:1" means the path begins with "/opt", and, and "36:0" means that the return value is 0. The mappings between the data values and the indices were set internally by a configuration file.

All of these features are important for different reasons. The time of day could indicate that the user is making system calls during normal business hours, or, alternatively, is logging in late at night, which could be anomalous. The path could indicate the security level of the system call being made – for instance, a path beginning with /sbin could indicate use of important system files, while a path like /bin/mail could indicate something more benign, like sending mail. The user ID is important to distinguish events, what is anomalous for one user may not be anomalous for another. A programmer that normally works from 9 A.M. to 5 P.M. would not be expected to login at midnight, but a maintenance technician (who performs maintenance on server equipment during off hours, at night), would. Frequent changes in machine IP address or changes that aren't frequent enough could indicate something anomalous. Lastly, the system call itself could indicate an anomaly most users would be expected to login and logout, but only administrators would be expected to invoke super user privileges with a command such as su.

We used LIBSVM [33] to build our models and to generate predictions for our test cases. First we will give an overview of our use of SVM software, which is standard procedure and is well documented in LIBSVM's help files. We chose to use the RBF (radial-based function) kernel for the SVM. It was chosen because it gives good results for our data set, but an in-depth search was not conducted to determine if this kernel is optimal. Parameters for the kernel (in the case of two-class SVM, C and gamma, and in the case of one-class SVM, nu and gamma) were chosen so that the F_1 measure was maximized. We chose to use the F_1 measure in this case (over other measures of accuracy) because, for the classifier to do well according to this metric, it must minimize false positives while also minimizing false negatives. Before training a model with our feature set, we used LIBSVM to scale the input data to the range [0, 1]. This was done to ensure that dimensions which takes on high values (like time) do not outweigh dimensions that take on low values (such as dimensions which represent categorical variables). The parameters that were used to scale the training data for the model are the same parameters that were used to scale that model's test data. Therefore, the model's test data will be in the vicinity of the range [0, 1], but it is possible that the test data goes outside of that range.

We conducted two experiments with the SVM. The first (experiment A) was designed to compare one-class SVM with two-class SVM for the purposes of insider threat detection, and the second (experiment B) was designed to compare a stream classification approach with a more traditional approach to classification. We will begin by describing our comparison of one-class and two-class SVM. For this experiment, we took the 7 weeks of data, and randomly divided it into halves. We deemed the first half "training" data and the other half "testing" data. We constructed a simple one-class and two-class model from the training data and recorded the accuracy of the model in predicting the test data.

The "stream" detection approach (EIT) is slightly more complicated in nature and thus merits a lengthier explanation. Algorithm 3 depicts pseudo code for the stream SVM process. We use an ensemble-based approach that is scored in real time. The ensemble maintains K models that use one-class SVM, each constructed from a single past day and weighted according to the accuracy of the models' previous decisions. For each test token, the ensemble reports the majority vote of its models.

Algorithm 3. Stream EIT Process

```
1.  Models  = new List()
2.  ModelAccuracies = new List()
3.  EnsembleAccuracy = new List()
4.  for each week W in Weeks:
5.      for each day D in W:
6.          Votes = new List()
7.          for each model M in Models:
8.              Predictions ← get the predictions of M for
                week W, day D
9.              Votes ← add Predictions, weighted with
                the ModelAccuracy of M, to Votes
10.         end for
11.         for each model M in Models:
12.             ModelAccuracies ← update Accuracies
                with the new accuracy of M, relative to
                the majority vote
13.         end for
14.             EnsembleAccuracy ← add the accuracy of
                the majority vote relative to the expected
                output to EnsembleAccuracy
15.             M ← train a new Model with data from
                week W, day D
16.             Add accuracy of M to ModelAccuracies
17.             Add M to Models
18.             if(Size of Models > K):
19.                 Remove model with least accuracy
                    from Models
20.             end if
21.     end for
22. end for
23. return EnsembleAccuracy
```

The stream approach outlined above is more practical for detecting insider threats because insider threats are stream in nature and occur in real time. A situation like that in the first experiment above is not one that will occur in the real world. In the real world, insider threats must be detected as they occur, not after months of data has piled in. Therefore, it is reasonable to compare our updating stream ensemble with a simple one-class SVM model constructed once and tested (but not updated) as a stream of new data becomes available (see Experiment B).

II. RESULTS

The performance of the ensemble e was measured by the accuracy of its predictions throughout 7 weeks of test data. The data was chosen because it is well documented with regard to actual anomalies at Lincoln Laboratory, and because of its large size, which should improve the accuracy of the results. Table 2 gives a summary of our results for experiment A.

One-class SVM outperforms two-class SVM in experiment A. Simply, two-class SVM is unable to detect any of the positive cases correctly. Although the two-class SVM does achieve a higher accuracy, it is at the cost of having a 100% false negative rate. By varying the parameters for the two-class SVM, we found it possible to increase the false positive rate (the SVM made an attempt to discriminate between anomaly and normal data), but in no case could the two-class SVM predict even one of the truly anomalous cases correctly. One-class SVM, on the other hand, achieves a moderately low false negative rate (20%), while maintaining a high accuracy (87.40%). This demonstrates the superiority of one-class SVM over two-class SVM for insider threat detection.

Table 2. Experiment A: One-Class vs Two-Class SVM

	One-Class SVM	**Two-Class SVM**
False Positives	3706	0
True Negatives	25701	29407
False Negatives	1	5
True Positives	4	0
Accuracy	.87	0.99
False Positive Rate	0.13	0.0
False Negative Rate	0.20	1.0

The superiority of one-class SVM over two-class SVM for insider threat detection further justifies our decision to use one-class SVM for our test of stream data. Table 3 gives a summary of our results for experiment B. The updating stream (EIT) achieves much higher accuracy than the non-updating stream, while maintaining an equivalent, and minimal, false negative rate (10%). The accuracy of the updating stream (EIT) is 76%, while the accuracy of the non-updating stream is 58%.

Table 3. Experiment B: Updating vs Non-updating stream approach

	Updating Stream	**Non-updating Stream**
False Positives	13774	24426
True Negatives	44362	33710
False Negatives	1	1
True Positives	9	9
Accuracy	0.76	0.58
False Positive Rate	0.24	0.42
False Negative Rate	0.1	0.1

Table 4. Description of the Dataset A

Entry	**Description—Dataset A**
User	Donaldh
# of records	189
Week	2-7 (Friday only)

Impact of Supervised and Unsupervised Learning: Now, our goal is to study the impact of learning, namely, supervised and unsupervised learning on true positive (TP), True negative (TN), false negative (FN) and false positive (FP). Here, we use smaller dataset as described in Table 4. We chose a

particular user (Donaldh) because anomaly happens only this user on week 6 at Friday. We have used the same ensemble size for both cases, K=3. For unsupervised learning we have used GBAD [42] with normative substructure q= 4.

Table 5. Experiment C:Supervised vs Non Supervised Learning approach

	Supervised Learning	Unsupervised Learning
False Positives	55	95
True Negatives	122	82
False Negatives	0	5
True Positives	12	7
Accuracy	0.71	0.56
False Positive Rate	0.31	0.54
False Negative Rate	0	0.42

The superiority of supervised learning over unsupervised learning for insider threat detection further justifies our decision to use supervised learning for our test of stream data. Table 5 gives a summary of our results for dataset A described in Table 4. For example, on dataset A the supervised learning achieves much higher accuracy (71%) than the unsupervised learning (56%), while maintaining lower false positive rate (31%) and false negative rate (0%). On the other hand, unsupervised learning achieves 56% accuracy, 54% false positive rate and 42% false negative rate.

Now, we turn our attention to varying the number of dimensions used by SVM on the accuracy of the stream ensemble. The number of dimensions can be varied by adding dimensions for attributes which were previously thought unimportant, or removing attributes which are less important. We can also vary the number of dimensions by changing the scope of certain categories. For instance, the path category for paths beginning with "/usr" can be made into more categories, by adding categories for subdirectories, such as "/usr/bin", "/usr/css", "/usr/sbin", etc. Figure 7 shows the effects of the number of dimensions on the accuracy of the ensemble.

The accuracy of the ensemble appears to decrease as dimensions are added, although the accuracy is highly sensitive to even small changes in the number of dimensions. We attribute this sensitivity to the fact that some dimensions are more important than others to the accuracy of the ensemble. For instance, in moving from 7 to 15 dimensions, the command attribute was added to the feature set. The command is an important attribute and adding it increased the accuracy of the ensemble. Adding some dimensions only confuses the ensemble. These dimensions do not allow the ensemble to effectively discriminate anomalous from normal data, and only blur the distinction. For instance, in moving from 36 to 47 dimensions, the user group attribute was added to the feature set. These dimensions only confused the ensemble as a user has the same user group when executing an anomalous command as when executing a normal one. The dimensions added last were the ones deemed least essential. Our results show that special consideration should be paid to the discriminatory power of each dimension. Unnecessary dimensions simply decrease performance.

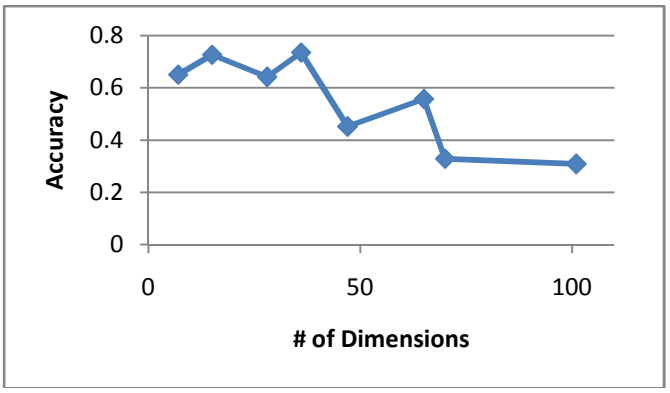

Figure 7. Accuracy of Ensemble vs # of Dimensions.

III. CONCLUSIONS AND FUTURE WORK

EIT performed well, with only one false negative and limited number of false positives. EIT is a combination of OCSVM and stream mining technology. EIT adopts advantages from both OCSVM & stream mining. In particular, OCSVM offered supervised learning and on the other hand stream mining offered adaptive learning. EIT was tested on MIT Lincoln lab dataset and shows effectiveness over OCSVM in terms of TP and FP.

We could extend the work in the following directions. First, further work could be made of testing EIT ensemble with varied numbers of dimensions and modeling different datasets to see if there is any way to more closely obtain a rate of zero false positives. Second, the polling algorithm could also be altered to create a voting system that contains more relevant votes. We will further enhance prediction accuracy strategy of each ensemble not only based on majority agreement but also some sophisticated methods. Finally, once a model is created in a supervised manner, we would like to update the model based on user feedback. Right now, once the model is created it remains unchanged. When ground truth is available over time, we would like to update the model further.

IV. ACKNOWLEDGMENT

This material is based upon work supported by the Air Force Office of Scientific Research under Award No. FA9550-08-1-0260. We thank Dr. Robert Herklotz for his support.

REFERENCES

[1] W. Eberle, L. Holder, and J. Graves, "Insider Threat Detection Using a Graph-Based Approach," *Journal of Applied Security Research*, vol. 6, Issue 1, January 2011.

[2] DARPA Intrusion Detectoin. (1998) Lincoln Laboratory: www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1998data.html

[3] SUBDUE: Graph Based Knowledge Discovery. (2003). http://ailab.wsu.edu/subdue

[4] M. Masud, J. Gao, L. Khan, J. Han, B. Thuraisingham, "A Practical Approach to Classify Evolving Data Streams: Traing with Limited Amount of Labeled Data," *Int. Conf. on Data Mining*, December 2008.

[5] A. Liu, C. Martin, T. Hetherington, and S. Matzner. "A Comparison of System Call Feature Representations for Insider Threat Detection." In *IEEE Information Assurance Workshop*, June 2005.

[6] S. Matzner and T. Hetherington, "Detecting Early Indications of a Malicious Insider," in *IAnewsletter*, vol. 7, 2004, pp. 42-45.

[7] N. Nguyen, P. Reiher, and G. H. Kuenning, "Detecting Insider Threats by Monitoring System Call Activity," presented at *Information Assurance Workshop*, 2003.

[8] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus, and Y. Vardi, "Computer Intrusion: Dectecting Masquerades," *Statistical Science*, vol. 16, pp. 58-74, 2001.

[9] K. Wang and S. Stolfo, "One Class Training for Masquerade Detection," *Workshop on Data Mining for Computer Security in conjunction with Third IEEE Conference Data Mining* , 2003.

[10] R. Maxion, "Masquerade Detection Using Enriched Command Lines," *International Conference on Dependable Systems & Networks (DSN-03)*, 2003.

[11] E. Schultz, "A Framework for Understanding and Predicting Insider Attacks," *Computers and Security*, vol. 21, 2002.

[12] 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*, vol. 6, D. Barbará and S. Jajodia, Eds. Boston: Kluwer Academic Publishers, 2002, pp. 77-102.

[13] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A Sense of Self for Unix Processes," *IEEE Symposium on Computer Security and Privacy*, 1996.

[14] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion Detection Using Sequences of System Calls," *Journal of Computer Security*, vol. 6, pp. 151-180, 1998.

[15] Y. Liao and V. R. Vemuri, "Using Text Categorization Techniques for Intrusion Detection," *11 USENIX Security Symposium*, 2002.

[16] E. Eskin, M. Miller, Z. Zhong, G. Yi, W. Lee, and S. Stolfo, "Adaptive Model Generation for Intrusion Detection Systems," *Workshop on Intrusion Detection and Prevention at the 7th ACM Conference on Computer Security*, 2000.

[17] C. Kruegel, D. Mutz, F. Valeur, and G. Vigna, "On the Detection of Anomalous System Call Arguments," *8th European Symposium on Research in Computer Security (ESORICS 2003),* Gjovik, Norway, 2003.

[18] G. Tandon and P. Chan, "Learning Rules from System Call Arguments and Sequences for Anomaly Detection," *ICDM Workshop on Data Mining for Computer Security (DMSEC)*, 2003.

[19] D. Gao, M. Reiter, and D. Song, "On Gray-Box Program Tracking for Anomaly Detection," *13th USENIX Security Symposium,* San Diego, CA, USA, 2004.

[20] "Understanding the Insider Threat: Proceedings of a March 2004 Workshop," *RAND National Defense Research Institute CF-196-ARDA*, July, 2004.

[21] L. Holder and D. Cook. Mining Graph Data. John Wiley and Sons, 2007.

[22] Hampton, M. and Levi, M. Fast spinning into oblivion? *Recent developments in moneylaundering policies and offshore finance centres. Third World Quarterly*, Volume 20, Number 3, June 1999, pp. 645-656, 1999.

[23] Eberle, W. and Holder, L. Mining for Structural Anomalies in Graph-Based Data, *International Conference on Data Mining*. June, 2007.

[24] Staniford-Chen, S. et al.. GrIDS – A Graph Based Intrusion Detection System for Large Networks. *Proceedings of the 19th National Information Systems Security Conference*, 1996.

[25] D. Cook and L. Holder. Graph-based data mining. *IEEE Intelligent Systems* 15(2):32-41, 1998.

[26] E. Kowalski, D. Cappelli, T. Conway, B. Willke, S. Keverline, A. Moore and M. Williams, "Insider Threat Study: Illicit Cyber Activity in the Government Sector," January 2008. URL: *http://www.cert.org/archive/pdf/insiderthreat_gov2008.pdf*

[27] Mohammad M. Masud, Qing Chen, Jing Gao, Latifur Khan, Charu Aggarwal, Jiawei Han, and Bhavani Thuraisingham. "Addressing Concept-Evolution in Concept-Drifting Data Streams". *ICDM '10: Proceedings of the IEEE International Conference on Data Mining (ICDM 2010)*, pages 929-934, Sydney, Australia, December 14-17, 2010.

[28] Mohammad M. Masud, Jing Gao, Latifur Khan, Jiawei Han, Bhavani M. Thuraisingham: Classification and Novel Class Detection in Concept-Drifting Data Streams under Time Constraints. *IEEE Trans. Knowl. Data Eng*. 23(6): 859-874 (2011).

[29] Wei Fan: Systematic data selection to mine concept-drifting data streams. *KDD 2004*: 128-137

[30] Pedro Domingos, Geoff Hulten: Mining high-speed data streams. *KDD 2000*: 71-80

[31] X. Yan and J. Han, "gSpan: Graph-based Substructure Pattern Mining," *IEEE International Conference on Data Mining*, 2002.

[32] Stolfo, Salvatore J. and Apap, Frank and Eskin, Eleazar and Heller, Katherine and Hershkop, Shlomo and Honig, Andrew and Svore, Krysta, "A comparative evaluation of two algorithms for Windows Registry Anomaly Detection", *Journal of Computer Security*, volume 13, issue 4, July, 2005, {issn, 0926-227}, pages 659-693, IOS Press, Amsterdam, The Netherlands.

[33] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[34] B. D. Davison and H. Hirsh. Predicting sequences of user actions. In Working Notes of the Joint Workshop on Predicting the Future: AI Approaches to Time Series Analysis, *15th National Conference on Artificial Intelligence/15th International Conference on Machine Learning*, pages 5-12. AAAI Press, 1998.

[35] Wen-Hua Ju and Yehuda Vardi. A hybrid high-order markov chain model for computer intrusion detection, *Journal of Computational and Graphical Statistics,* June, 2001.

[36] [R. A. Maxion and T. N. Townsend. Masquerade detection augmented with error analysis, IEEE Transactions on Reliability, 53(1):124{147, 2004.

[37] M. Schonlau, W. Dumouchel, W. Ju, A. F. Karr, M. Theus, and Y. Vardi. Computer intrusion: Detecting masquerades. Statistical Science, 16:58--74, 2001.

[38] Boleslaw K. Szymanski and Yongqiang Zhang. Recursive data mining for masquerade detection and author identification. *In proceedings of the 13rd Annual IEEE Information Assurance Workshop, IEEE Computer Society Press*, 2004.

[39] K. Wang and S. J. Stolfo. One-class training for masquerade detection. *In Proceedings of the 3rd IEEE Workshop on Data Mining for Computer Security*, 2003.

[40] Malek Ben Salem, Shlomo Hershkop, Salvatore J. Stolfo. "A Survey of Insider Attack Detection Research" in *Insider Attack and Cyber Security: Beyond the Hacker*, Springer, 2008

[41] Mohammad M. Masud, Jing Gao, Latifur Khan, Jiawei Han, Bhavani M. Thuraisingham: A Practical Approach to Classify Evolving Data Streams: Training with Limited Amount of Labeled Data. *ICDM 2008*: 929-934

[42] Pallabi Parveen, Jonathan Evans, Bhavani Thuraisingham,Kevin Hamlen, Latifur Khan, Insider Threat Detection Using Stream Mining and Graph Mining, to appear in The *Proc. of Third IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT-2011)*, Oct 2011, MIT, Boston, USA

[43] Larry M. Manevitz , Malik Yousef, One-class svms for document classification, *The Journal of Machine Learning Research,* 2, 3/1/2002

[44] V.N. Vapnik, An overview of statistical learning theory, *IEEE Trans. On Neural Networks*, vol. 10, issue 5, 1999, pp. 988-999.