Insider Threat Detection in PRODIGAL

Henry G. Goldberg, William T. Young, Matthew G. Reardon, Brian J. Phillips, and Ted E. Senator*
*Leidos, Inc., Arlington, VA USA (*former employee)*
{goldberghg, youngwil, reardonmg, phillipsb}@leidos.com; t.senator@verizon.net

***Abstract*—This paper reports on insider threat detection research, during which a prototype system (PRODIGAL)[1] was developed and operated as a testbed for exploring a range of detection and analysis methods. The data and test environment, system components, and the core method of unsupervised detection of insider threat leads are presented to document this work and benefit others working in the insider threat domain.**

We also discuss a core set of experiments evaluating the prototype's ability to detect both known and unknown malicious insider behaviors. The experimental results show the ability to detect a large variety of insider threat scenario instances imbedded in real data with no prior knowledge of what scenarios are present or when they occur.

We report on an ensemble-based, unsupervised technique for detecting potential insider threat instances. When run over 16 months of real monitored computer usage activity augmented with independently developed and unknown but realistic, insider threat scenarios, this technique robustly achieves results within five percent of the best individual detectors identified after the fact. We discuss factors that contribute to the success of the ensemble method, such as the number and variety of unsupervised detectors and the use of prior knowledge encoded in detectors designed for specific activity patterns.

Finally, the paper describes the architecture of the prototype system, the environment in which we conducted these experiments and that is in the process of being transitioned to operational users.

***Index Terms*—Anomaly detection; insider threat; unsupervised ensembles; experimental case study**

I. Introduction

Malicious insiders are adversarial and may attempt to hide their actions by employing techniques that they believe will evade detection. As in other adversarial domains, a useful insider threat detection system must be able to detect not only instances of known, suspected, or hypothesized insider threat scenarios, but also instances of previously unseen and novel insider threat scenarios [21].

Insider threat detection can be more difficult than other similar domains such as money laundering, intrusion detection, or counter-terrorism because insiders may be more aware of an organization's information protection capabilities and procedures than outsiders. Furthermore, malicious insider activity is typically only a small fraction of the activities performed by any user of an organization's information systems. We will see in the next section that this low "signal to noise ratio" is reflected in the test data, as is the unknown nature and diversity of threat scenarios.

The scenarios presented in this paper draw inspiration from case study literature of documented insider threat incidents from the public and private sectors ([6], [14], [18]). These examples of insider threat behaviors generalize across insider threat detection applications. Understanding the types of attacks designed to steal protected information informs the development of defensive enterprise detection technologies such as decoys ([4], [17]). Understanding complex human behaviors in enterprise environments supports the identification of patterns of activity in computer usage data related to behaviors associated with insider threat actions, such as quitting ([10], [5]). Characterizing behavior demonstrated by users in online social communities such as deception ([3]) and negative predisposition toward law enforcement ([13]) can be relevant to the insider threat domain. Machine learning techniques, such as outlier detection [2] and unsupervised anomaly detection [7], have likewise shown utility for identifying insider threat activiites in large, complex data sets ([9], [24]). Finally, the efficacy of ensembles of detectors has been shown ([1], [8]) and inspired our approach.

This paper concludes more than five years of work on insider threat detection during which a prototype system (PRODIGAL) was developed and operated to provide a testbed for exploring a wide range of detection and analysis methods.

Since this paper continues and extends experiments previously reported in [25], we have repeated some introductory material and descriptions of our approach - especially regarding metrics, the test data, and the ensemble approach. While the previous paper looked at results from data collected over eight months, we cover results from 16 months, including a number of novel inserted target scenarios. We also present the PRODIGAL system architecture and discuss its utility in potential layered applications. Finally, we are able to draw stronger conclusions about ensemble performance and to suggest future directions.

The first phase of the program was devoted to developing the data structures and detection algorithms needed to operate a large suite of detectors. The two main foci of the work performed during the second phase of the program involved approaches to support analysts in a real operational system.

- The fact that analysts need a single ranked list of subjects for investigation led to experiments with combining multiple detector results to produce a single set of high-quality leads.
- The need to provide analysts with more information about the component decisions comprising a lead inspired methods to provide explanations of detector output and other (temporal and aggregate) analyses to support down stream decision makers. [12]

We present findings from experiments to detect instances of insider threat scenarios inserted into a real database from monitored activity on users' computers seeded with independently-

[1]PROactive Detection of Insider Threats with Graph Analysis and Learning

URI: http://hdl.handle.net/10125/41476
ISBN: 978-0-9981331-0-2

developed and inserted insider threat activities superposed on the activities of real users and inserted into the underlying system to be monitored in the same way as any other user activities[2]

The rarity of known targets, as well as the diversity of threat scenarios provided in the test data preclude our ability to make strong claims of repeatability or robustness. However, we discuss findings from exploratory experiments applying a diverse suite of anomaly detection methods and compare them individually with the output of an unsupervised ensemble combination method.

- We describe an unsupervised ensemble-combination technique whose performance, when operating on outputs from a large diverse set of anomaly detectors, is close to that of the best (post hoc) detector over many months of data and multiple scenario types.
- We see that the same unsupervised ensemble-based anomaly detection technique outperforms detectors having features designed specifically for individual suspected scenarios.
- Initial comparisons of the best performing anomaly detectors with those selected in the final step of the ensemble method show interesting potential for improving the method.
- We note instances where individual detectors and the ensemble, although performing poorly on some scenarios vis. an overall metric like AUC[3], can still support the ability to detect leads to detection of complex insider threat scenarios involving unknown groups of actors collaborating over days or weeks.

Experiments and results included in this paper extend previously reported results ([21], [24], [25]) to cover 16 months of data from September 2012 through February 2014. Testing on the additional eight months included a number of new Red Team (RT) scenarios, new detection algorithms, and improved versions of existing detectors.

Finally, this paper describes PRODIGAL's architecture and discusses some of the design issues and tradeoffs that affect how to deploy an insider threat detection system incorporating these methods.

II. Background: Data, Scenarios and Detectors

A. Test Data and Red Team Scenarios

Test data for experimentation consists of a database of 5,500 users. The data collection system, SureView®[4], records all user behaviors for specified activities, such as logon/logoff, email, file actions, instant message, printer, process, and URL events for a calendar month. On average, there are 1,000 events per user per active day. Data are made available on a monthly basis. The data provider anonymizes all user identification (ID) and other personally identifiable information (PII) in the data set and hashes all information related to user events to a randomly generated but internally consistent designator.

Separately from the data collection process, an independent Red Team develops scenarios reflecting their field experience of threat behaviors. Scenarios encapsulate specific insider threat actions that are superimposed on the actions of real users identified as appropriate to particular roles in the scenario. The Red Team inserts up to five instances of scenarios (with variations) in a data month for a total of 54 instances of twenty-eight distinct scenarios. ([6], [11], [23])

The Red Team, in an effort to avoid evaluation bias, designs its scenarios independently of our detection methods. Likewise, we do not review scenario specifics nor train our detectors on the test data to avoid over-fitting. Scenario descriptions are provided after the fact for purposes of evaluating our methods; the Red Team is continually adding new scenarios as the research is ongoing. Neither the Red Team nor our team claims that the set of scenarios is complete.

B. Developing a Diverse Suite of Detectors

PRODIGAL is constructed to enable a variety of experiments in insider threat detection over the test data previously described. The key techniques have been described in detail in [21], [24] and [25].

Our core hypothesis is that a diverse suite of detectors, coupled with a method for combining their results, will be sufficient to produce high quality leads for further investigation, either by human analysts or automated reasoning components.

In this section, we describe a diverse suite of detectors of three types: (I) indicator-based, (A) anomaly-based, and (S) scenario-based. Table I lists the detectors, configured in PRODIGAL and reported in this paper.

Indicator-based detectors use statistical outlier techniques applied to sub-sets of features related to one or two particular types of activity such as file or web access. They are most like the detectors aimed at "tells" or specific activities, which are typically used in current practice.

Anomaly detectors employ complex models that focus on different aspects of the data, e.g. structural features, semantic features, or temporal features, and then search through the entire feature space to identify potential anomalies. Features typically consist of observed actions, aggregates, or ratios, such as URLs accessed by a user, the number of print jobs by a user, or the ratio of the number of files copied to removable media compared to the total number of files actions. Relational features such as the email and text-message communication graphs are used to provide comparison groups in some detectors. Different approaches to feature normalization are incorporated into variants of the same detection models used in PRODIGAL [21].

Scenario-based detectors are designed to detect specific patterns of activity known to be associated with some insider threat scenarios. They are inspired by the scenarios described in [6], but are developed independently of the Red Team scenario descriptions and inserts. They consist of a combination of indicator-based and anomaly-based detectors and classifiers in a specified workflow, structured to reflect a hypothesized combination of real world actions that are likely to discriminate

[2]The information in this tool is collected from users from an unspecified organization. All data is used with permission, in a closed facility subject to all necessary privacy protections.

[3]area under the receiver operating characteristic (ROC) curve

[4]See https://www.forcepoint.com.

TABLE I: User-Day Detectors in PRODIGAL

Algorithm	Type	Description
TBAD	A	Temporal Based Anomaly Detection
VSM	A	Vector Space Models
GFADD:1	I	Grid-based Fast Anomaly Detection with Duplicates (GFADD) (File creation : distinct file no grid)
GFADD:2	I	GFADD (Distinct file count : files on remov. count grid:8)
GFADD:3	I	GFADD (Copies to : from removable drives no grid)
GFADD:4	I	GFADD (Copies to : from removable drives grid:8)
GFADD:5	I	GFADD (File : removable drive events grid:8)
GFADD:6	I	GFADD (Fixed event count : network event count no grid)
GFADD:7	I	GFADD (Fixed event count : network event count grid:8)
GFADD:8	I	GFADD (Fixed : removable event counts grid:8)
GFADD:9	I	GFADD (Network events : distinct removable drives no grid)
GFADD:10	I	GFADD (Removable drive events : distinct no grid)
GFADD:11	I	GFADD (Removable drive events : distinct grid:8)
GFADD:12	I	GFADD (Removable drive events : network events no grid)
GFADD:13	I	GFADD (Removable drive events : network events grid:8)
GMM:RD	A	Gaussian Mixture Model (GMM) (Raw count features user-day scores)
GMM:QD	A	GMM (Quantile features)
EGMM:RD	A	GMM (Raw count features)
EGMM:QD	A	Ensemble GMM (Quantile features)
CROSS:RD	A	Cross Prediction (CP) (Raw count features)
CROSS:(QD	A	CP (Quantile features)
RIDE:RD	A	Repeated Impossible Discrimination Ensemble (RIDE) (Raw count features)
RIDE:QD	A	RIDE Ensemble (Quantile features)
IFOR:RD	A	Isolation Forest (IFor) (Raw count features)
IFOR:QD	A	IFor (Quantile features)
CADE:R	A	Classifier-Adjusted Density Estimation (CADE) (Raw features)
CADE:UP	A	CADE (UP features)
PDE:R10K	A	Pseudo-likelihood Density Estimator (PDE) (raw features 10k training)
PDE:UP10K	A	PDE (UP features 10k training)
PDE:UP	A	PDE (UP feature set)
Saboteur	S	Scenario: Saboteur (Variant 2)
IP Thief	S	Scenario: IP Thief (Variant 1)
Fraudster	S	Scenario: Fraudster (Variant 1)
Amb. Lead.	S	Scenario: IP Thief Ambitious Leader (Variant 1)
File	I	Indicator: File Activity (Variant 1)
URL	I	Indicator: URL Activity (Variant 1)
File-URL	I	Indicator: File vs. URL (Variant 1)
URL,File-Log.	I	Indicator: URL and File vs. Logon
Careless	S	Scenario: Carless User
Rager	S	Scenario: Rager

between the scenario of interest and other, mostly legitimate, actions. Six scenario-based detectors have been deployed in varying stages of development in PRODIGAL. They focus on particular sub-spaces of features that are relevant to a particular scenario, as well as sub-sets of target users and/or time periods, as suggested by the scenario. In this way, domain knowledge of both activity type and relevant comparison peer groups are incorporated into the detection. They are:

- Saboteur: An insider uses corporate information technology (IT) resources to harm an organization or an individual.
- IP Thief: An insider uses corporate IT resources to steal intellectual property.
- Fraudster: An insider mis-uses IT for for personal gain or to commit a crime.
- Ambitious Leader: The Ambitious Leader recruits other insiders to get access to all parts of the IP being stolen.
- Careless User: The insider is not intentionally malicious but disregards corporate IT policies, exposing the organization to risk.
- Rager: The insider has outbursts of strong or threatening language in Email/Webmail/IM coinciding with anomalies in other activities, indicating a potential fundamental change in behavior.

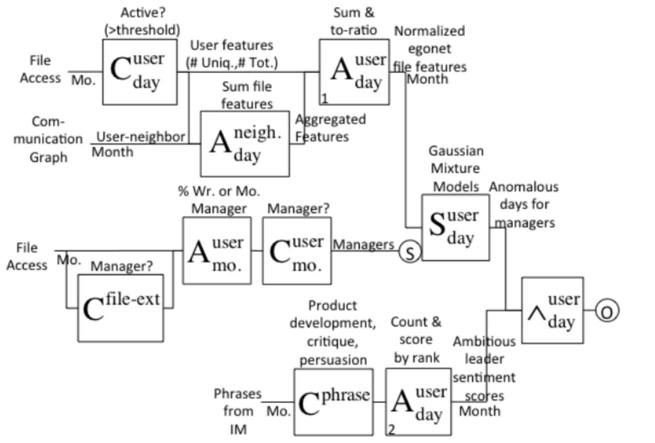


Fig. 1: ADL Diagram of IP Thief Ambitious Leader Scenario

A particular detector specification may incorporate a statistical model of normal behavior, a set of features derived from user activities, a baseline population for comparison (i.e., a peer group), a time period for the baseline activity, time granularity for potential detection, a particular approach to feature normalization, and other relevant aspects (for example, user job categories). Baselines for comparison may be cross-sectional (i.e., compare a user's actions over a particular time period with that of other users in a peer group over some time comparable time period) or temporal (i.e., compare a user with his/her own behavior over different time periods), or both. Especially for the more complex, scenario detectors, we develop specifications using an Anomaly Detection Language (ADL) that was introduced in [21] and [24]. We have found specification of a complex planned detection in ADL to be extremely useful in the process of developing and testing combinations of feature sub-setting, classification, anomaly detection, and threat ranking mechanisms. Figure 1, taken from [24], shows the design of a complex scenario-based detector.

III. Detector Ensemble Performance

As we discussed earlier, the primary role of the PRODIGAL prototype is to enable exploratory experiments in the insider threat space. This reflects the goals of the ADAMS program generally, and is an effective way to advance the state of practice in this domain.

In the following sections, we present results from running and evaluating the detector ensemble over the full two years of live data + Red Team inserts. We begin with the choice of performance metrics and discuss the potential contribution of PRODIGAL's anomaly detection approach to overall analysis of insider threats. We then describe the ensemble method used to produce a single threat ranking from the detector suite. Then we discuss the performance of the ensemble itself, its component detectors, and how they relate to one another.

A. Measuring Overall Detector Performance

Section II.A. above describes the data environment in which our prototype operates. Instances of Red Team scenarios are limited to one month duration and inserted as targets each calendar month. (The Computer Emergency Response Team (CERT), a division of Carnegie-Mellon's Software Engineering Institute, has found that 2/3 of known insider threat scenarios evolve over less than one month.) This allows for consistent, independent experiments. However, because of the (realistic) rarity of threat inserts and the variety of scenarios, there is very little repeated data with which to validate robustness or measure sensitivity of our methods to different levels of threat activity. Furthmore, since we do not have ground truth regarding the live data, we cannot measure the true precision, recall, or false alarm rate, only the rates of detection vis. Red Team inserts.

The experiments reported here measure detection performance on *user-day* entity extents, a data structure derived from the collection of activities of one user over one day. (We limit entity extents to this size for these experiments, although PRODIGAL is capable of representing many others.) We consider a hit to be the ranking of a user-day above some threshold, and can measure the hit/false-alarm tradeoffs using measures of the rate of true positives and false negatives.

B. Addressing the Needs of Insider Threat Surveillance

Metrics were chosen to measure both detection accuracy of the individual algorithms and their contribution to the overall task of providing leads to an analyst. For the former, we compute the Receiver Operator Characteristic (ROC) curve and area under the curve (AUC) as well as the Approximate Lift Curve and Average Lift. AUC estimates discrimination, or the probability that a randomly chosen positive entity extent will be ranked higher than a randomly chosen negative one.

The choice of metric is critical to achieve the goal of providing workable leads to an analyst. Initial development has relied on the AUC metric, and, for simplicity, that is what is reported here. However, AUC may be less suitable for a highly asymmetrical detection situation, where a very few positives must be identified high in the ranking to support an effective layered detection process in which analysts receive at least one lead from each (or most) scenarios near the top of the list and can then 'connect the dots' to the rest of the malicious behavior.

Lift metrics, such as Average Lift, estimate the improvement in target density delivered to later stages of a multi-stage detection process. We also compute the number of positive

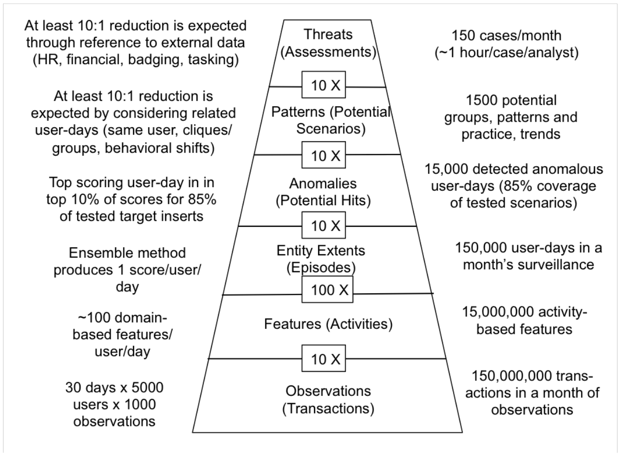

At least 10:1 reduction is expected through reference to external data (HR, financial, badging, tasking)
At least 10:1 reduction is expected by considering related user-days (same user, cliques/ groups, behavioral shifts)
Top scoring user-day in in top 10% of scores for 85% of tested target inserts
Ensemble method produces 1 score/user/ day
~100 domain-based features/ user/day
30 days x 5000 users x 1000 observations
Threats (Assessments)
10 X
Patterns (Potential Scenarios)
10 X
Anomalies (Potential Hits)
10 X
Entity Extents (Episodes)
100 X
Features (Activities)
10 X
Observations (Transactions)
150 cases/month (~1 hour/case/analyst)
1500 potential groups, patterns and practice, trends
15,000 detected anomalous user-days (85% coverage of tested scenarios)
150,000 user-days in a month's surveillance
15,000,000 activity-based features
150,000,000 transactions in a month of observations

Fig. 2: Feasibility of Insider Threat Detection based on Anomaly Detection

hits ranked in the top k scored entity extents (for k = 5, 10, 50, 100, 500, etc.) and in the top p% of all scored entity extents (for p = .01, .05, .1, .5, 1, 5, etc.). The latter allow us to estimate anticipated detection success for fixed analyst workloads.

We are investigating other lift-focused approaches, aiming to improve the detection of leads to better support a layered detection process (as described in [22]). Figure 2 depicts how such processes, individually with very modest lift values, can be combined to solve the problem. The unsupervised anomaly detection and ensemble methods we describe subsequently in this paper can achieve the indicated performance. Further research we are pursuing with domain-knowledge based pattern detection and threat assessment will be able to provide further lift to support insider threat surveillance on an ongoing basis in an organization.

The following sections report on our investigation of unsupervised anomaly detection, ensemble methods for combining diverse detection results, and the role that domain-specific detectors may or may not play in the acquisition of leads for insider threat surveillance.

IV. Unsupervised Ensemble-based Anomaly Detection

A. Methods

An analyst responsible for insider threat detection needs a single ranked list rather than many different result sets from different detectors whose detailed operation he/she may not fully understand. Anomaly detector ensemble methods combine the results (i.e., scores) from multiple detectors in a way that is analogous to how classifier ensembles combine predictions from multiple classifiers [8]. The following description of our ensemble approach is taken from [25]. We have not significantly altered the algorithm for the experiments reported here.

Selecting an approach for building ensembles depends upon the types of detectors that are used. If all the detectors share an underlying model, then the ensemble approach can leverage that commonality to improve performance, e.g., the method reported in [15] varies the features used as input to a single anomaly detection model to build an ensemble. Another way of leveraging a common model is to use the same input features, but alter hyperparameters, which determine how the model is built in each detector ([15], [8]).

Because our individual detectors employ a variety of models, we chose an approach, presented in [19], that is consistent with such a diverse ensemble of detectors. This method employs two heuristics:

1) If a consensus about which points are most anomalous can be drawn from the individual detectors, then that consensus should be preserved in the final ensemble.
2) Because each individual detector is subject to unavoidable biases stemming from the choice of model, choice of input features, hyperparameter settings, etc., the ensemble should prefer combinations of results from detectors with uncorrelated biases.

These heuristics are implemented in two distinct phases in this method. In the first phase they extract a consensus across all detectors from the union of the top k most-anomalous points from each detector. All points in this union are given a score equal 1.0 and all others are given a score of zero. We chose a value for k for each dataset that included the top 1.0% of the points. The method then initializes the ensemble with scores from the detector that is most correlated with the consensus. The correlation between detectors and the consensus is found by viewing each as n-length vectors of scores, where there are n points in the dataset, and then using a simple correlation metric to compare the vectors. We used the Pearson's *r* correlation metric for this.

In the second phase, the method selects candidate detectors to combine with the initial ensemble, preferring detectors that are least correlated with the current ensemble. The same correlation metric used before is used again here. The candidate detector's scores are combined with the current ensemble using a point-wise combination function. For this the method uses the average over scores for each point; we also experimented with other functions including the maximum of scores. The algorithm proceeds to accept a candidate detector if the resulting ensemble is no less correlated with the consensus than the previous ensemble; if it is, then the candidate detector is discarded. This phase continues until all detectors are either accepted or discarded.

B. Results and Discussion

Our initial experiments evaluated our detection results based on our ability to detect user-days with red-team inserted activity. We used a wide variety of detectors, described in [21], and the ensemble technique described above. Results are summarized in Table II. For each month, we report the area under the ROC curve for our best detectors, for the ensemble, and the ratio of the two, indicating how close the ensemble came to the best. The AUCs reflect the ability of the detectors to find all of the user-days of the union of all scenarios present in the month. These results illustrate that the unsupervised ensemble-based anomaly-detection technique had performance that is close to that of the best of the individual anomaly detectors. The AUC for the ensemble technique was within five percent of the AUC of the best detector, with a 95% confidence interval of +/- 1.5% around a linear regression fit of ensemble to best AUC over 54 individual RT scenario instances. Interestingly, the ensemble-based technique appeared to have results that were similar across datasets, while the best detector varied widely.

Figure 3 illustrates the differences in performance between the ensemble and the best-performing detector for selected months. These eight months have been selected to show instances when a wide variety of detectors were best performer.

In September and October 2012 (Figure 3a and Figure 3b) there were six instances of the same RT scenario (14-17, 20-21), and one of the CADE detectors was best both times. Although neither CADE nor the Ensemble were able to score all inserted user-days very high, all inserted scenarios had several user-days in very high rankings, supporting the expectation that a later stage of analysis would easily find the rest given these leads.

Figure 3c through Figure 3e and Figure 3g show months where different detector models performed best. In each case the Ensemble was able to approximate performance of the best detector. These months contained 12 instances of 10 different RT scenarios.

Figure 3f shows Ensemble performance significantly lower than the best detector, although the top few user-days are ranked highly by both. However this detector, PDE:UP, averaged only 85% of the Ensemble's AUC values over all 16 months, and so could not be relied upon for consistent detection. This is the case for all detectors of all three types and is the principal reason why even an ensemble method that fails to improve on the best score is still valuable.

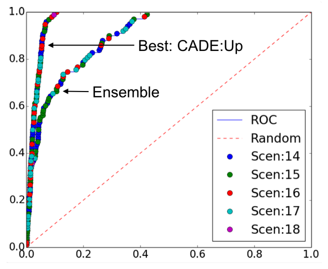
Finally, we note for the month shown in Figure 3h, the best detector – GFADD with no grid over the feature pair, *# file events on removable drives* vs. *# distinct removable drives* – only returns a score for user-days it finds anomalous, so the value of an AUC is questionable. The second best detector is also shown. Again, the Ensemble comes close, in fact exceeding its AUC value slightly.

We are particularly interested in how well the scenario-based detectors perform, since they contain domain knowledge. We see that the ensemble generally out-performed the scenario-based detectors, including the scenario-based detectors that we determined later to have been a likely fit to the Red Team scenario that was actually inserted. In Table III we see AUC performance on individual inserted scenario instances. Ensemble AUC over these instances averages 0.87 and consistently outperforms the selected relevant scenario-based detector; in three of the 41 by over 150%. There is considerable variation in the results, even over instances of the same RT scenario, suggesting the inherent variability of the RT threat simulation process.

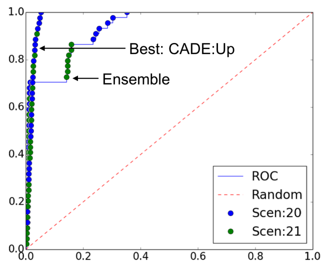
Table III reports results on all months where the comparison was possible, not only months where the ensemble performed well. Figure 4 depicts the performance of the ensemble method compared to the corresponding scenario-based detectors for four of these months. Figure 4a shows a month where the inserted scenarios and the scenario-base detectors correspond

TABLE II: Ensemble and Best Detector Results by Month

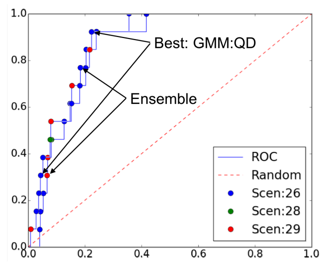
Month	Ensemble AUC	Best Detector	Best Det. AUC	Ens. / Best	RT Scenarios
12-Sep	0.8973	CADE:UP	0.9703	92.47%	Circumventing SureView Insider Startup
12-Oct	0.9319	CADE:UP	0.9804	95.05%	Insider Startup
12-Nov	0.7542	File	0.7895	95.53%	Anomalous Encryption, Layoff Logic Bomb, Masquerading 2
12-Dec	0.8646	GMM:QD	0.8677	99.64%	Anomalous Encryption, Layoff Logic Bomb, Outsourcer's Apprentice
13-Jan	0.8594	RIDE:RD	0.9015	95.34%	Hiding Undue Affluence, Outsourcer's Apprentice, Survivor's Burden
13-Feb	0.7632	EGMM:QD	0.7793	97.94%	Bona Fides, Manning Up, Survivor's Burden
13-Mar	0.8853	IFOR:QD	0.8963	98.77%	Bona Fides, Hiding Undue Affluence, Manning Up Redux
13-Apr	0.8635	RIDE:QD	0.8619	100.19%	Circ. SureView, Indecent RFP, Selling Login Cred., Survivor's Burden
13-May	0.8469	PDE:UP	0.9718	87.14%	Credit Czech, Exfiltration Prior to Termination
13-Jun	0.8852	IFOR:QD	0.9103	97.24%	Czech Mate, Exfiltration of Sensitive Data Using Screenshots
13-Jul	0.8498	RIDE:RD	0.8769	96.90%	Breaking the Stovepipe, Snowed In
13-Oct	0.8938	GMM:RD	0.8972	99.62%	Breaking the Stovepipe, Snowed In
13-Nov	0.8479	RIDE:RD	0.8459	100.23%	Byte Me, Naughty by Proxy
13-Dec	0.8034	EGMM:RD	0.828	97.02%	Byte Me Middleman, Indecent RFP 2, Passed Over
14-Jan	0.8425	IFOR:RD	0.8242	102.22%	From Belarus With Love, Passed Over, What's the Big Deal
14-Feb	0.847	GFADD:84-88-0	0.9775	86.65%	Bollywood Breakdown, Breaking the Stovepipe, Gift Card Bonanza, Naughty by Proxy



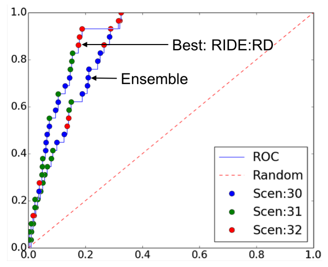
(a) September 2012



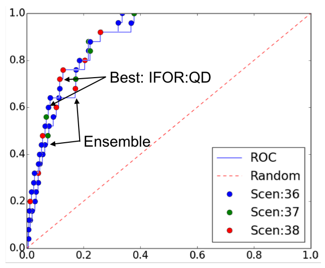
(b) October 2012



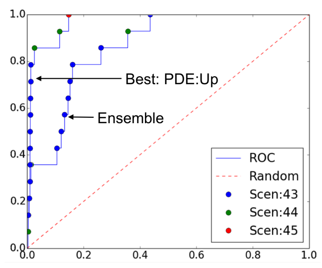
(c) December 2012



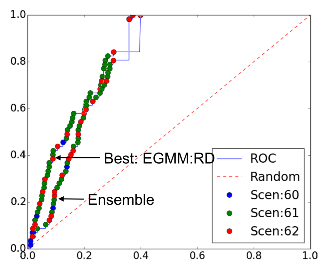
(d) January 2013



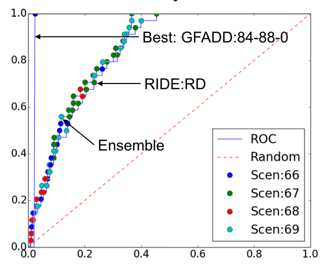
(e) March 2013



(f) May 2013



(g) December 2013



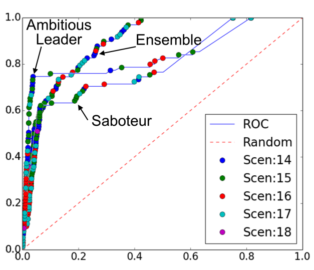
(h) February 2014

Fig. 3: ROC curves vs. all RT inserts for the Ensemble and the best detector for various months.

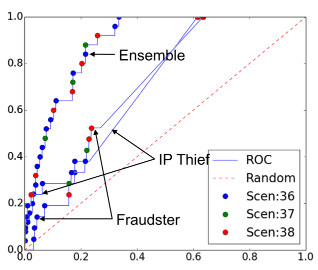
fairly well. The typical response of the scenario-based detectors is to score some of the best matching user-days well, often better than the Ensemble, but then drop off rapidly in the remainder of inserted targets. IP Thief in Figure 4b through Figure 4d shows this behavior, although the presence of other scenarios in the month masks its performance on the relevant inserts. We have reviewed individual scenario-based detectors on separate inserted RT scenarios and see the same response, but the data are too cumbersome to present graphically in this paper. However, it is worthwhile to note the number of instances in Table III where scenario-based detectors performed close to the Ensemble cases where the detector does well on a few high ranking user-days at the expense of not identifying others. We are investigating ways to make use of these targeted responses.

Figure 5 identifies the sets of detectors selected by the ensemble each month and compares them to the best performing detectors for that month. The best-performing detector was included in the ensemble in only four of the 16 months of data. And because in those months there are on average more than six detectors selected for the final scoring step, and all ensembles comprise equally-weighted detectors, the best detector is never given more than one sixth of the weight in this ensemble result. Therefore, the ensemble technique is able to achieve comparable performance to the best detector by combining detectors and with those detectors often excluding the best-performing detector. Recall that the heuristics driving the ensemble technique favor detectors that are either most close to the consensus or those that are able to add diversity to the ensemble (least correlation with the ensemble) without reducing correlation with the consensus. Thus in these data sets the best-performing detector generally disagrees with the consensus from other detectors, yet a combination of those other detectors can be built automatically that performs nearly as well as that best detector.

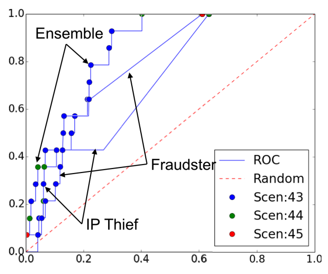
In eight out of 16 months at least one of the accepted



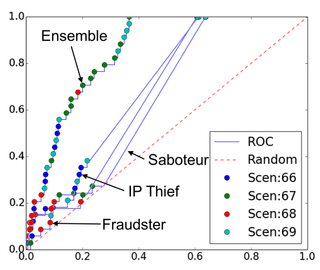
(a) September 2012



(b) March 2013



(c) May 2013

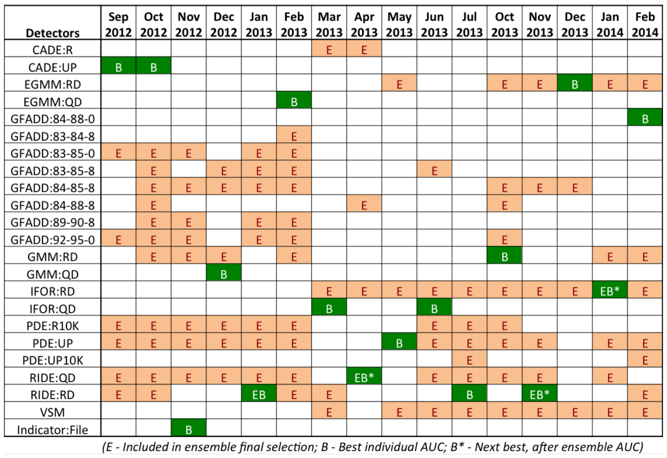


(d) February 2014

Fig. 4: ROC curves for the Ensemble and the Scenario-Based Detectors for various months.

TABLE III: Comparison of Scenario-Based Detector to Ensemble Performance, by Red Team Scenario Inserts

Month	Inserted RT Scen.	Ens. AUC	Relevent Scenario Detector	Scen. Det. AUC	Ratio
12-Sep	Insider Startup	0.87	Amb. Lead.	0.92	95%
12-Sep	Insider Startup	0.88	Amb. Lead.	0.69	128%
12-Sep	Insider Startup	0.93	Amb. Lead.	0.92	102%
12-Sep	Insider Startup	0.91	Amb. Lead.	0.84	108%
12-Sep	Circumventing S.V.	0.98	Saboteur	0.95	103%
12-Oct	Insider Startup	0.92	Amb. Lead.	0.93	99%
12-Oct	Insider Startup	0.95	Amb. Lead.	0.82	115%
13-Mar	Manning Up Redux	0.90	IP Thief	0.76	119%
13-Mar	Hiding Undue Aff.	0.85	Fraudster	0.82	104%
13-Mar	Bona Fides	0.84	IP Thief	0.66	129%
13-Apr	Survivor's Burden	0.88	Saboteur	0.69	128%
13-Apr	Selling Login Cred.	0.86	Amb. Lead.	0.62	139%
13-Apr	Indecent RFP	0.84	Fraudster	0.65	128%
13-May	Credit Czech	0.83	Fraudster	0.86	97%
13-May	Exfil. Pre-Termination	0.85	IP Thief	0.81	105%
13-May	Exfil. Pre-Termination	1.00	IP Thief	1.00	100%
13-Jun	Exfil. w/Screenshots	0.81	IP Thief	0.47	173%
13-Jun	Exfil. w/Screenshots	0.99	IP Thief	0.98	102%
13-Jun	Exfil. w/Screenshots	0.93	IP Thief	0.47	197%
13-Jun	Czech Mate	0.87	Fraudster	0.60	145%
13-Jul	Breaking the Stovepipe	0.82	IP Thief	0.80	102%
13-Jul	Snowed In	0.86	Fraudster	0.72	118%
13-Oct	Snowed In	0.86	Fraudster	0.83	105%
13-Oct	Snowed In	0.93	Fraudster	0.69	134%
13-Oct	Snowed In	0.87	Fraudster	0.81	107%
13-Oct	Breaking the Stovepipe	0.93	IP Thief	0.84	111%
13-Nov	Naughty by Proxy	0.79	Saboteur	0.57	138%
13-Nov	Naughty by Proxy	0.72	Saboteur	0.49	147%
13-Nov	Byte Me	0.86	Fraudster	0.72	119%
13-Nov	Byte Me	0.89	Fraudster	0.74	120%
13-Dec	Indecent RFP 2	0.94	Fraudster	0.72	131%
13-Dec	Byte Me Middleman	0.83	Fraudster	0.62	133%
13-Dec	Passed Over	0.75	Saboteur	0.50	150%
14-Jan	Passed Over	0.86	Saboteur	0.66	130%
14-Jan	What's the Big Deal	0.90	Careless	0.69	130%
14-Jan	From Belarus w/Love	0.72	IP Thief	0.63	115%
14-Feb	Bollywood Breakdown	0.93	IP Thief	0.79	117%
14-Feb	Gift Card Bonanza	0.77	Fraudster	0.64	120%
14-Feb	Breaking the Stovepipe	0.95	IP Thief	0.84	114%
14-Feb	Naughty by Proxy	0.81	Saboteur	0.57	141%



Detectors	Sep 2012	Oct 2012	Nov 2012	Dec 2012	Jan 2013	Feb 2013	Mar 2013	Apr 2013	May 2013	Jun 2013	Jul 2013	Oct 2013	Nov 2013	Dec 2013	Jan 2014	Feb 2014
CADE:R							E	E								
CADE:UP	B	B														
EGMM:RD									E			E	E	B	E	E
EGMM:QD						B										
GFADD:84-88-0																B
GFADD:83-84-8						E										
GFADD:83-85-0	E	E	E		E	E										
GFADD:83-85-8		E		E	E	E				E						
GFADD:84-85-8		E	E	E	E	E						E	E	E		
GFADD:84-88-8		E						E				E				
GFADD:89-90-8		E	E		E	E										
GFADD:92-95-0	E	E	E		E	E						E				
GMM:RD		E	E	E		E						B			E	E
GMM:QD				B												
IFOR:RD							E	E	E	E	E	E	E	E	EB*	E
IFOR:QD							B			B						
PDE:R10K	E	E	E	E	E	E				E	E	E				
PDE:UP	E	E	E	E	E	E			B	E	E	E	E		E	E
PDE:UP10K											E					E
RIDE:QD	E	E	E	E	E	E		EB*		E	E	E	E		E	
RIDE:RD	E	E			EB	E	E				B		EB*			E
VSM							E		E	E	E	E	E	E	E	E
Indicator:File			B													

(E - Included in ensemble final selection; B - Best individual AUC; B - Next best, after ensemble AUC)*

Fig. 5: Ensemble Composition and Best-Performing Detector, by Month

detectors used an underlying model that was shared with the best-performing detector. For example, in December 2012 the best performing algorithm is GMM Density Estimation via unusualness of counts vs company, which shares the same underlying model Gaussian mixture models as one of the accepted detectors, GMM Density Estimation using Raw Counts; the difference between these two detectors is the method of normalizing input features, which we mentioned as an important element of detector configuration and a source of diversity in our detection suite.

These apparently contradictory results, good performance of the ensemble without participation of the best detector in the final scoring step, have led us to devise a series of simulation experiments to thoroughly understand the role of the participating detectors in ensemble methods. It is clear that the best detector plays a role in driving ensemble performance. When the best detectors perform poorly, so does the ensemble. So their role in constructing the consensus must be important. We aim to improve ensemble performance significantly through this research by applying the two heuristics more effectively as well as through continued improvement of the individual detectors.

V. Complex Scenario Detection

The threat scenarios that form the targets for detection typically involve complex inter-related sets of small numbers of actions by multiple individuals with varying relationships occurring over multiple days to weeks. Their actions are superposed on normal activities.

In contrast, cyber threats are typically executed by software at time scales of seconds or less. Network-based monitoring is typically used to detect such cyber threats. Here, we rely on host-based monitoring of user actions. The essential challenge

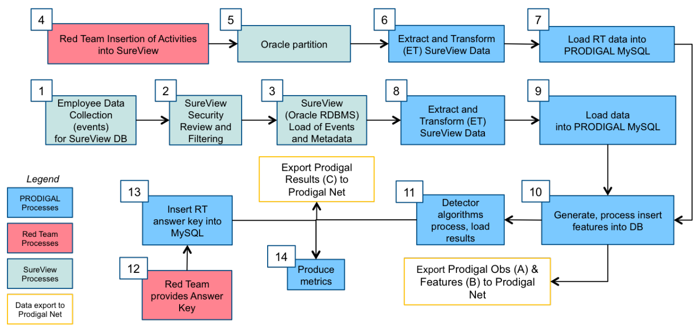

4
Red Team Insertion of Activities into SureView
5
Oracle partition
6
Extract and Transform (ET) SureView Data
7
Load RT data into PRODIGAL MySQL
1
Employee Data Collection (events) for SureView DB
2
SureView Security Review and Filtering
3
SureView (Oracle RDBMS) Load of Events and Metadata
8
Extract and Transform (ET) SureView Data
9
Load data into PRODIGAL MySQL
Legend
PRODIGAL Processes
Red Team Processes
SureView Processes
Data export to Prodigal Net
13
Insert RT answer key into MySQL
Export Prodigal Results (C) to Prodigal Net
11
Detector algorithms process, load results
10
Generate, process insert features into DB
12
Red Team provides Answer Key
14
Produce metrics
Export Prodigal Obs (A) & Features (B) to Prodigal Net

Fig. 6: Monthly Pipeline Processing in Testbed Environment

to detect these actions is that the effective signal-to-noise ratio is dependent not only on the fraction of actions by the malicious individuals that are improper but also on the combinatorics of the grouping of individuals.

To illustrate PRODIGAL's ability to detect such complex scenarios, we repeat a discussion of one such target scenario inserted in multiple data set by the Red Team as described in [25].

The Insider Startup scenario for the month of October 2012 involved two separate instances each of three distinct users who engaged in malicious activity on 22 user-days on 12 distinct calendar days in a calendar month. The first of these instances comprised 527 distinct observations of user activity, of which 151 involved file accesses, 19 involved text messages, 42 involved processes, and 12 involved URL accesses. The second of these instances involved 468 inserted observations, of which one was an email event, 148 were file events, 41 process events, 18 keyboard events, and 278 URL events. These 527/468 inserted events correspond to roughly 0.5-0.5-0.6% of each user's total activity over the course of a month, or about 0.0003% of the total observations in the database of about 5500 users each month. The scenario was designed with minimal communications between the three malicious users to make it realistically difficult to detect; the key challenge was to aggregate behavior to detect any of the conspirators. The highest-ranked user-day from our best detector for the two instances of this scenario; i.e. the number of user-days that would have to be reviewed by an analyst to detect this malicious behavior, was 17 and 13 out of the approximately 150,000 user-days analyzed that month.

Similar results were achieved on four instances of the same scenario in September 2012, and on an instance of the Indecent RFP scenario in April 2013, where the highest ranked user-day was involved in the target and thus could provide a lead for subsequent analysis.

VI. Prototype Architecture

PRODIGAL operates in the research environment as a manually controlled pipeline as illustrated in Figure 6. In this research environment, it is executed monthly to correspond with the Red Team scenario insertion and evaluation processes. As new versions of components are developed, tested, and considered for incorporation in the prototype, different versions and/or configurations of some modules are executed for experimental purposes; these execution sequences usually involve all components downstream of the subject of the experiment. For potential production environments, PRODIGAL can be configured to execute more frequently and/or on different data periods (e.g., weekly execution on a rolling four-week period). This section describes the PRODIGAL components and the controlling software framework that enables this variety of different execution methods, beginning with the details of the components in the context of the monthly testbed processing that generated the majority of the results reported in this paper.

A. *Monthly Pipeline Processing in Testbed Environment*

The PRODIGAL prototype ingests user activity data on a daily or monthly basis. Figure 6 shows a series of data processing stages (identified by a number). The PRODIGAL prototype executes each of those stages in series within a pipeline structure.

Processing begins with monitoring of events by SureView on user workstations and organizational servers. (boxes 1-3) Information collected by SureView is warehoused in an Oracle database at the SureView server and is used for regular monitoring by security personnel. A copy of the data is transmitted to the ADAMS testbed environment, where it is anonymized by removal or hashing of personally identifying information (PII) and stored in an instance of the SureView warehouse schema. In parallel, the Red Team creates additional SureView events (boxes 4-5) that are inserted into a separate partition for merger with the collected data.

PRODIGAL processing begins with an ETL component (box 8) that extracts data from the Oracle SureView database and (box 9) transforms and loads it into the PRODIGAL schema in a MySQL database. The purpose of this transformation is to convert the data from SureView observations into user activities. The same ETL processes (boxes 6 and 7) are executed on the real user data and on the Red Team insert data.

The real and Red Team data are merged using table views, and PRODIGAL loads this combined data into its MySQL database. This area of the PRODIGAL database is referred to as the PRODIGAL Observation Store. PRODIGAL next computes the features that serve as the basis for its detectors and augments the PRODIGAL Observation Store with these computed features (box 10).

Detectors, consisting of algorithms and their associated parameterizations, create anomaly scores for all user-days in the data set (box 11). Each detector separately scores each user-day, so there are many scores for each user-day. These scores are stored in a separate MySQL database called the PRODIGAL Results Store, and are indexed by algorithm id value, user id value, and a sequential run id value. The algorithm id specifies the exact algorithm used to generate a score. Algorithms read in many feature values from a user, a time period, or a population to assess how anomalous a behavior set is. The PRODIGAL prototype uses a user id as a unique identifier for a specific person operating a computer.

The next step in the processing flow, also included in box 11, is the execution of the ensemble algorithm, which produces the official single score for each user-day. A user with a very

TABLE IV: Stages and Longest Run Times

Stages	Performance Time
ET (6) and Run Time Load (7) of simulated Red Team data	1 day
ET (8) and Load (9) of source data from SureView database	10 days
Generate features (10)	2 days
Run Algorithms	7 days

high anomaly score represents a candidate for further security investigation activities. The Red Team provides an answer key to the team after the pipeline is run (stage 12). These labels are inserted into the database (box 13) and used to determine how accurate the output scores were and to compute the detection metrics discussed earlier in this paper (box 14).

The ADAMS testbed environment interacts with two other environments; 1) a research and development environment in which algorithm experimentation can occur on derived data and statistical summaries, and 2) an operational test environment in which the real data are processed (without the Red Team inserts) for evaluation by security personnel, with feedback on the highly-scored user-days to be provided back to the research team. Each environment is composed of multiple computers with different hardware configurations and source data sets.

Researchers use the research and development environment to create new framework and algorithm software. It is housed within a development network that is separate from all other data sources. It receives data exports from the testbed environment, and acts as a development platform used by data analysts to explore results using novel and experimental processes. An engineering team uses this to develop framework code to manage the overall PRODIGAL prototype. Algorithm designers use this to experiment with algorithms, test approaches, and create production quality anomaly detection algorithms within a test environment.

The operational test environment accepts data from the same real data feeds as the production environment. It contains a PRODIGAL prototype instance that performs daily ingestion of user data, generates feature scores, and computes algorithm and ensemble scores on a weekly basis using a rolling four-week baseline. This environment has no simulated insider threats. The operational test environment represents PRODIGAL outside of its research role, focusing on how it will be used in the real world'. Configuring the operational test environment provided engineers with experience that will be useful to enable deployment of PRODIGAL to actual customer environments and to demonstrate its utility to security analysts on their own real world datasets. PRODIGAL components are installed in the operational test environment using RedHat's Package Manager (RPMs).

All framework source code development and unit testing takes place in the research and development environment. In the event that some source code is developed in other environments, it must also be included in the research and development environment. The development environment contains several git repositories for different components used for source code version control. Researchers upload software into an environment using either source code (for later Java compilation) or an automated RPM installation process. Engineers automate the creation of RPM packages by using a Maven plug-in component.

Source code development encompasses different computer languages and approaches. Framework software developers use the Java programming language. Algorithm developers and analysts use a variety of computer languages. These analysts contribute skills and software libraries to be used within the overall PRODIGAL prototype. Algorithm components have been built using Java, C, C++, Steel Bank Common LISP, Python, and R. The framework triggers these algorithms through a command line call. Framework developers have created a set of Java classes used to wrap a command line call within a Java object. Spring framework context files integrate these diverse applications together into a single system. The wrapper classes launch applications that use different programming languages, pass database connection information to these algorithm components, and detect when processes have finished.

The framework uses individual wrapper classes to create an asynchronous flow of algorithms. A flow is a list of wrappers annotated with dependency information. A flow starts one algorithm, waits for it to finish, then starts the next. The total number of asynchronous algorithms is dependent on the available processors, and the dependencies described within the flow.

B. *Performance*

Each stage of the pipeline requires time and computing resources. Most of the time used to execute the ETL and the detection algorithms. Table IV shows the pipeline stages that consume the most processing time on a 40 CPU server with 512 GB of RAM. The team measured the performance times in Table IV by executing a monthly pipeline using the testbed environment.

VII. Conclusions and Future Work

In this paper, we evaluate PRODIGAL in the setting of complex, adversarial insider threats inserted into a database of real user activities. These threat are unknown and varied in composition, methods, and goals. We show that by using a variety of diverse individual detectors combined using an anomaly detection ensemble technique, we achieve a final detection result with performance that consistently approaches that of the best detector on each dataset (in after-the-fact analysis).

This result holds on many threat data sets, including ones following scenarios we had not contemplated when designing the detectors. The ensemble result also outperforms many anomaly detectors that are specifically focused on the scenarios that are known, on data sets containing those scenarios or scenarios with similar behaviors. Furthermore, we investigate the composition of the ensembles chosen by our technique and find that the ensemble achieves consistent performance without relying any single detector or the best unidentified detector for each dataset.

We extend preliminary results (from [25]) to an additional eight months, including 27 additional threat data sets comprising 15 new Red Team scenarios. The average AUC achieved by the ensemble over the new data is identical to that over the initial data, at 0.85, while the average "best" detector AUC only improved slightly from 0.88 to 0.89. These observations suggest that our approach is robust to gaps between the scenarios contemplated during detector design time and unexpected scenarios that appear in real data, so long as the available detectors are still diverse and numerous as we have in our prototype.

This result is one that we hope to study in future work. Specifically, we are interested in developing more advanced ensemble techniques than the one we used that are able to incorporate scenario-based detectors effectively to increase confidence in results when known scenarios do match with ones in the data. We have also begun incorporating explanation capabilities with the ensemble approach so that underlying reasons for detection from individual detectors can be combined in the final result presented to analysts. These explanations are made available to the analysts in a newly designed user interface.

Acknowledgment

The work reported herein was performed as part of the Anomaly Detection at Multiple Scales (ADAMS) program sponsored by the Defense Advanced Research Projects Agency (DARPA). We thank our collaborators at Oregon State University, University of Massachusetts, Georgia Institute of Technology, and Carnegie Mellon University, and Leidos who developed many of the algorithms that were used for detection and who developed and operate the software that configures and executes these algorithms in the ADAMS testbed. We also thank the ADAMS Red Team from CERT who develops the scenarios, inserts them into the real data, and provides the answer keys against which we are evaluated. Finally, we thank the data provider and testbed operator for the ADAMS program for making the real data available for research and for operating the testbed in which it occurs. Funding was provided by the U.S. Army Research Office (ARO) and DARPA under Contract Number W911NF-11-C-0088. The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] C. Aggarwal, Outlier Ensembles, In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, ODD '13*, New York, ACM, 2013.

[2] C. Aggarwal, Outlier analysis, In *Data Mining*, (pp. 237-263), Springer International Publishing, 2015.

[3] L. Akoglu, R. Chandy, C. Faloutsos, Opinion Fraud Detection in Online Reviews by Network Effects. *ICWSM*, 13, 2-11, 2013.

[4] B. Bowen, et. al., Monitoring Technologies for Mitigating Insider Threats. In *Insider Threats in Cyber Security*, C. Probst, et. al. (Eds.) (pp. 197217), New York: Springer US, 2010.

[5] O. Brdiczka, et. al., Proactive insider threat detection through graph learning and psychological context. In *Security and Privacy Workshops (SPW)*, 2012 IEEE Symposium on Security and Privacy (pp. 142-149), IEEE, 2012.

[6] D. Cappelli, A. Moore, R. Trzeciak, *The CERT Guide to Insider Threats: How to Detect, Prevent, and Respond to Information Technology Crimes*, Addison-Wesley Professional, 2012.

[7] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey. *ACM computing surveys (CSUR)*,41(3), 15, 2009.

[8] T. Dietterich. Ensemble Methods in Machine Learning, In *Multiple Classifier Systems*, 115. Springer, 2000.

[9] H. Eldardiry, et. al., Multi-domain information fusion for insider threat detection. In *Security and Privacy Workshops (SPW)*, (pp. 45-51). IEEE, 2013.

[10] G. Gavai, et. al., Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 6(4), 47-63, 2015.

[11] J. Glasser, B, Lindauer, B., Bridging the gap: A pragmatic approach to generating insider threat data. In *Security and Privacy Workshops (SPW), 2013 IEEE* (pp. 98-104), IEEE, 2013.

[12] H. Goldberg, et. al., "Explaining and Aggregating Anomalies to Detect Insider Threats", in *Proceedings 49th Hawaii International Conference on System Sciences (HICSS)*, Koloa, HI, USA, Jan. 5-8, 2016.

[13] M. Kandias, et. al., Proactive insider threat detection through social media: The YouTube case. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society* (pp. 261-266), ACM, 2013.

[14] E. Kowalski, et al., Insider threat study: illicit cyber activity in the government sector, United States Secret Service & the Software Engineering Institute, Carnegie Mellon University, January 2008.

[15] A. Lazarevic and V. Kumar, Feature Bagging for Outlier Detection, In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 157166, 2005.

[16] F. T. Liu, K. M. Ting, and Z. H. Zhou, Isolation Forest, In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM'08)*, Pisa, Italy, 15-19 December 2008, pp. 413422.

[17] Y. Park, S. Stolfo, Software decoys for insider threat. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security* (pp. 93-94). ACM, 2012.

[18] M. Randazzo, et. al., *Insider threat study: Illicit cyber activity in the banking and finance sector (No. CMU/SEI-2004-TR-021)*. Carnegie-Mellon University Software Engineering Institute, 2005.

[19] E. Schubert et. al., On Evaluation of Outlier Rankings and Outlier Scores, in *Proceedings of the 12th SIAM International Conference on Data Mining (SDM)*, Anaheim, CA, 2012, 10471058, 2012.

[20] E. Schultz, A framework for understanding and predicting insider attacks. *Computers & Security*, 21(6), 526-531, 2002.

[21] T. E. Senator et. al., Detecting Insider Threats in a Real Corporate Database of Computer Usage Activity, in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 1393-1401, ACM (2013).

[22] T. E. Senator, On the Efficacy of Data Mining for Security Applications, in *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*, Paris, France, June 28, 2009.

[23] K. Wallnau, et. al., "Simulating malicious insiders in real host-monitored user data", In *Proceedings of the 7th USENIX conference on Cyber Security Experimentation and Test*, pp. 4-4. USENIX Association, 2014.

[24] W T. Young et. al., Use of Domain Knowledge to Detect Insider Threats in Computer Activities, in *Proceedings of the Workshop on Research for Insider Threat*, IEEE CS Security and Privacy Workshops, San Francisco, CA, 23-24 May 2013.

[25] W T. Young et. al., Detecting Unknown Insider Threat Scenarios, in *Proceedings of the Workshop on Research for Insider Threat*, IEEE CS Security and Privacy Workshops, San Jose, CA, 18 May 2014.