Use of Domain Knowledge to Detect Insider Threats in Computer Activities

William T. Young, Henry G. Goldberg, Alex Memory, James F. Sartain, Ted E. Senator
SAIC
Arlington, VA, USA
{youngwil, goldberhg, memoryac, sartainjf, senatort}@saic.com

***Abstract*— This paper reports the first set of results from a comprehensive set of experiments to detect realistic insider threat instances in a real corporate database of computer usage activity. It focuses on the application of domain knowledge to provide starting points for further analysis. Domain knowledge is applied (1) to select appropriate features for use by structural anomaly detection algorithms, (2) to identify features indicative of activity known to be associated with insider threat, and (3) to model known or suspected instances of insider threat scenarios. We also introduce a visual language for specifying anomalies across different types of data, entities, baseline populations, and temporal ranges. Preliminary results of our experiments on two months of live data suggest that these methods are promising, with several experiments providing area under the curve scores close to 1.0 and lifts ranging from x20 to x30 over random.**

Keywords-anomaly detection; insider threat; experimental case study

I. INTRODUCTION

The goal of the Defense Advanced Research Projects Agency's (DARPA's) Anomaly Detection at Multiple Scales (ADAMS) program is to combine structural and semantic information from a real corporate database[1] of monitored activity on users' computers to detect independently developed red team (RT) inserts of malicious insider activities. Over the past two years, the PRODIGAL (**PRO**active **D**etection of **I**nsider Threats with **G**raph **A**nalysis and **L**earning) research team[2] has developed 16 anomaly detection algorithms based on various models of behavior, including vectors of activity features, temporal sequences of events, and graph models of relationships among users and computer resources. In addition, we developed data extract, transform and load components and an integration framework that enables rapid integration of algorithms with features extracted from the data for experimentation. It operates in a closed laboratory setting that is dedicated to the ADAMS program. In this environment, the real corporate database has synthetic insider activity inserted into it by a red team led by CERT, that bases the inserts on realistic threat scenarios [11].

This paper presents the results of the PRODIGAL team's efforts to detect anomalies indicative of insider threats (ITs) based on structural and semantic features of the data. It focuses on the application of domain knowledge to provide starting points for further analysis and on experimental results, rather than the details of the anomaly detection algorithms. It also presents a new anomaly detection language used to specify instances of anomaly detection processes. The experimental results are analyzed according to multiple robust metrics and are based on two months' worth of test data from September and October of 2012. To our knowledge, this is the first set of results from a comprehensive set of experiments to detect realistic IT instances in a real corporate database of computer usage activity.

Most prior research has focused on countering malicious insider behavior from a law enforcement or information security perspective. References [2][3][5][6][7] contain significant domain knowledge drawn from workshops, surveys, and case studies of ITs. Beyond lower-level intrusion detection, or policies and procedures to reduce risk, there is limited previous work on detecting IT in real databases. A comprehensive survey of IT detection methods [9] discusses methods for modeling user behavior to detect masquerading, profiling activities in Windows® Microsoft Corp. and Web environments, and methods to integrate detection with more active approaches for entrapping malicious actors. Graph-based analysis has been investigated extensively to model social and information patterns in the context of IT scenarios. Eberle et al [4] presents several scenarios and graphical models for their detection.

Our research differs from prior work by incorporating domain knowledge into multiple starting points for analysis in the form of *a priori* indicators of threat, anomaly detection across multiple models and data types, and high-level pattern detection driven by known or suspected threat scenarios. We find that this combination of semantic and structural analysis across multiple scales, i.e., from low-level transactions to high-level patterns of behavior, detects instances of realistic IT scenarios – consisting of complex combinations of activities – with high accuracy.

II. PROVIDING STARTING POINTS FOR ANALYSIS

We apply domain knowledge to develop three types of starting points for analysis of observable computer usage activity that suggest known or suspected ITs. The types of starting points are indicators, anomalies, and scenarios. Each type incorporates domain knowledge at a different

[1] The database is from a large corporation whose identity is not allowed to be disclosed publicly. All data are used with permission in a closed facility subject to all necessary privacy protections.

[2] The PRODIGAL research team consists of SAIC, Georgia Tech, Oregon State University, University of Massachusetts and Carnegie Mellon University.

DOI 10.1109/SPW.2013.32

TABLE 1. STAGES OF INSIDER THREAT SCENARIOS

Stage of Activity	Goal: Destruction of information or systems	Goal: Misuse or corruption of information or systems	Goal: Theft of information or systems
Exploration	Locate weak points	Locate access points	Locate data
Experimentation	Trial insertions	Trial modifications	Trial access
Exploitation	N/A	N/A	Staging data
Execution	Plant malware	Modify data	Exfiltrate data
Escape & Evasion	Shift blame	Cook books	Layer movement

level. Indicators are models of user behavior found in computer usage observables that are known to correlate with malicious insider actions as derived from case studies (e.g., increased removable media activity may be indicative of intellectual property (IP) theft). Anomalies focus on unusual patterns in the data (e.g., logins after hours, large numbers of file events on network drives). Here, we draw on an understanding of human-computer interactions and how abnormal patterns from a malicious insider might occur in computer usage observables within single or multiple feature sets over different time ranges. Scenarios describe complex patterns of malicious insider actions that span data types (e.g., email, file access, login, and URL) and entity, population, and temporal extents and baselines. Scenarios we used are suggested by case studies of real insider attacks on government and commercial systems [3][6][7]. Note that we defined and used these scenarios without any knowledge of the scenarios inserted by the red team.

III. USE OF DOMAIN KNOWLEDGE

A. Designing a Feature Space from Domain Knowledge

In designing models for malicious insider detection, we first consulted with a retired operations officer from the U.S. Intelligence Community. We did this to focus our subject-matter expertise on the activities of the malicious insiders themselves – recruiting, tasking, deploying, obfuscating – with the goal of understanding how to detect these activities.

We model malicious insider behavior according to goals and stages of threat activity. We identified three main goals: (1) destruction of information or systems; (2) misuse or corruption of information or systems; or (3) theft of information or systems. We then categorized malicious insider actions into five stages: (1) exploration; (2) experimentation; (3) exploitation; (4) execution; and (5) escape and evasion. Table 1 summarizes these activities.

These goals and stages of malicious activity can be adapted to cover a range of possible behavior such as malicious insiders acting alone or in groups (with complicit and non-complicit members in those groups) that exhibit multiple behaviors across various temporal extents (e.g., day, week, and month).

We then derived features from base computer usage observables involved with the activities identified in Table 1. Table 2 lists the number of features by activity type and gives some examples of each type. Note that the final group is a set of ratios, which we believe provide features normalized by the insider's own activity levels.

TABLE 2. ACTIVITY-BASED FEATURES

Type	#	Examples
Email	18	Count of attachments on sent emails
File	28	Count of file events to removable drives
Group	11	Shared printers
Login	4	Count of distinct workstations logged onto
Printer	9	Count of print jobs submitted
URL	13	Count of Blacklist events
Ratio	28	Ratio of file events on removable drives to all file events Ratio of URL uploads to URL downloads Ratio of distinct removable drives to URL upload/download events

B. Indicators of Malicious Activity

When activity of a particular type described above is unusual in a way or to a degree known to correlate with malicious insider actions, we can treat these features as *a priori* indicators. An example is the File Events indicator, which scores all user activity for all file features (28) over the temporal extent (user day) and produces a single score related to unusually high file access and movement. While not all such episodes are malicious, there is sufficient evidence from case studies to warrant further analysis. More than one activity type may be combined, as with File + URL actions, which may correlate with information exfiltration over the web.

C. Anomaly Detection Algorithms

Because PRODIGAL's components must also find unknown ITs, our team experimented extensively with unsupervised anomaly detection (AD) algorithms using the features we have discussed above, as well as relationship and temporal sequence data derived from the base cyber observables. These led to a number of AD algorithms. We report on some of the most successful below.

D. Complex Scenarios of Inter-related Activities

In addition to the feature-driven experiments involving indicators, we constructed several scenario detectors from documented cases of malicious insider behavior [3][6][7] as well as from our prior experience. Detectors were

implemented in the PRODIGAL framework using available features, indicators, and outlier detection algorithms, as well as peer groups discovered by graph-based community detection algorithms using graphs of shared computer resources.

Saboteur: An insider's use of information systems to direct specific harm at an organization or an individual. Saboteurs are technical, such as a system administrator, have privileged access to systems, and revenge is generally their motivation. They set up their attack before termination and execute the attack after leaving.

Intellectual Property (IP) Thief-Ambitious Leader: An insider's use of information systems to steal IP from the organization. This category includes industrial espionage involving insiders. IP thieves are generally scientists, engineers or salespeople. They generally steal what they consider to be their own work from their former or current project and use it to start a new company or give it to a new company or foreign organization. We present a detailed description of detecting this scenario later in this paper.

Intellectual Property (IP) Thief-Entitled Individual (EI): An IP thief that steals information like the EI, but is motivated by ambition to steal as much as possible before leaving the organization, rather than dissatisfaction or a dispute. To do so, he recruits other insiders to get access to all parts of the IP being stolen.

Fraudster: An insider's use of information systems for the unauthorized modification, addition, or deletion of an organization's data (not programs or systems) for personal gain, or theft of information that leads to an identity crime (e.g., identity theft, credit card fraud). Fraudsters are lower-level employees often motivated by financial need – hardship, greed, etc. They sometimes are recruited by outsiders in collusion with other insiders.

Careless User: The insider is not intentionally malicious but, through blatant disregard of corporate policies concerning information systems systems, exposes the group to a comparable level of risk (i.e., compromising systems and data) similar to the Saboteur scenario.

Rager: The insider has outbursts of strong, vociferous, abusive, and threatening language in email/Webmail/instant messages (IM) repeatedly toward other insiders or against the organization in general . These outbursts coincide with anomalies in other data types, e.g., Logons, URL, indicating a potential fundamental change in behavior.

IV. Anomaly Detection Language

Specifying the analytics and domain knowledge for the three types of starting points requires combining multiple methods applied to different baseline and peer group populations over distinct time periods. For example, we may want to detect users (or collaborating groups of users) whose daily behavior over a recent month differs from their daily behavior over a previous six-month period with respect to themselves or to their peers in the same work group or job role. Traditional data flow diagrams cannot express these designs concisely, so we developed a visual anomaly detection language that enables the expression of such combinations of methods, data, baselines, and detection extents. While developed for IT detection, the language itself is domain-independent and may be applied to other domains. The language specifies the extent of the entities to be detected (e.g., individual users or groups of users) combined with the temporal extent of potential anomalies. Inputs to these expressions are transactional records of user activity, and outputs are scores on these user-temporal extents.

The syntax of the language is shown in Figure 1; required arguments are in <angle brackets> and optional arguments in [square brackets]. Records are passed along horizontal lines from left to right. Component types are specified by symbols. Entity and temporal extents are super- and sub-scripts, respectively, of component type.

Anomaly detector components may be statistical (denoted by the symbol S) or temporal (T); the latter indicating detectors specialized for anomalies in temporal patterns. Group detectors (G) discover communities of entities, which can be used as baseline populations. Classifiers (C) place input records into classes, which may also be used as baseline populations, or for filtering or partitioning records in general. The classes may be hard, meaning that each record is put into exactly one class, or mixed, in which case a record may be a member of more than one class, possibly to varying degrees. Classifiers might be implemented using a machine-learning method or may be a simple filter based on a lookup on a record. Similarly, aggregators (A) group records with some shared characteristic and summarize their values, e.g., roll-up emails from the same sender to a single record having the count of emails as a new feature; aggregators derive new features from existing ones in this way. Another way to transform features is with a normalizer (N), e.g., rescale real-valued features to the unit interval. Finally, if given a baseline, records are classified and normalized with respect to that baseline.

When sets of records are joined and contain different

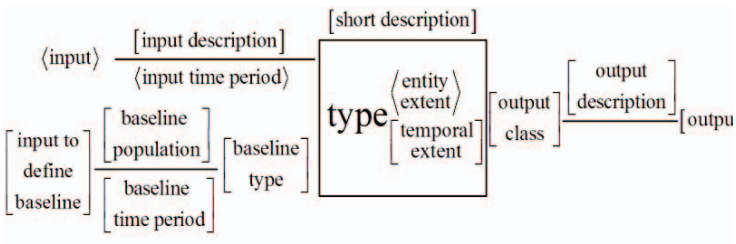

Figure 1. Anomaly Detection Language Syntax

values for the same feature, ***and*** and ***or*** (⋀, ⋁) can combine those values, e.g., implement with a t-norm and t-conorm to combine unit-interval values. Evidence combiners (E) also combine values but are more general than ⋀ and ⋁. And, when no combinations are necessary, union (∪) and intersection (∩) perform the expected operations on input records.

If a baseline is provided, a baseline type specifies how the baseline is used by the component and is indicated by a symbol inside a small circle to the left of the component to which the baseline input connects. With a cross-sectional baseline (C), entity extents are compared to others within the same temporal extent. In contrast, with a longitudinal baseline (L) each entity will be handled individually, and different temporal extents for that entity are compared to one another. A simultaneous baseline (S) combines the first two and compares each input extent to all baseline extents. If a baseline or input time period is not specified, this means that the two cover all available time periods.

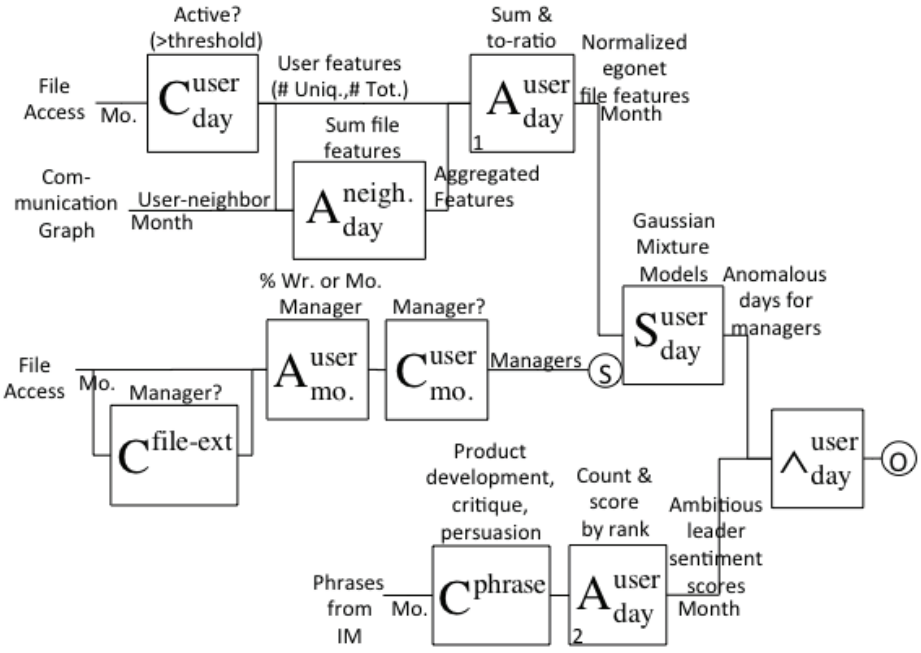

Figure 2. IP Thief – Ambitious Leader Scenario Diagram

Whenever a component may output more than one output class of records, e.g., a binary classifier has (+) and (−) output classes, they should be placed to the right of the component inside circles connected to output lines, unless only one class of output is needed and that class is clear from context, in which case the output class can be omitted.

Weights are scalars in the unit interval used to transform features – usually scores – and are drawn as the letter w inside a rectangle. The type of weighting should be put in a description above the rectangle. Finally, the output of the system is drawn as the letter "O" inside a circle.

In Section V.C.1, we use this language to describe an example scenario in detail.

At present, we manually translate detection specifications into workflow descriptions that are submitted to a flow controller that automatically orchestrates services that execute the algorithms in the specification against the specified data. We are currently working to automate the manual translation step and develop an interactive builder interface for detection specification.

V. EXPERIMENTS

Experiments in the PRODIGAL Framework have been aimed at exploring the space of potential starting points for IT analysis. Given a stable, realistic background of several months of computer observables over a population of 5500 full-time users and ground truth consisting of inserted red team target activities, we ran a wide variety of indicator detectors, anomaly detection algorithms, and scenarios. We applied several performance metrics to try to understand the potential effectiveness of each detector or detection method, as well as of the whole suite, for IT detection. The following sections discuss our findings.

A. Test Data

Experiments featured data collected using an instance of SureView® (Raytheon Oakley Systems, Inc.) [10] in an organization that consisted of approximately 5,500 users. All user identification (ID) and personal information were anonymized and hashed to a unique user ID number for each user. Only computer-based user activity was visible and all activities were related to the hashed user IDs. Test data included login, file, printer, browsing, IM, process, and email events involving these users. On average, users had approximately 1000 actions per day with totals varying by data type.

To provide ground truth in the test data, an external, independent red team inserted insider activity based on several threat scenarios into the test data. For the experiments discussed in this paper, the principal red team scenario consisted of three insiders who colluded over IM and corporate email to steal IP and form a new company. The first insider entices the other two to find sensitive technical information (e.g., files) and copy that information to removable media for exfiltration outside the network. The red team ran two variants of this scenario over consecutive,

two-months' worth of test data, inserting a total of six instances. [11]

B. Metrics

We utilize several metrics for this stage of our experiments, with the following assumptions:

- Each method (indicator, AD algorithm, or scenario) scores, and thus ranks, a number of entities (treated as trials in a binary classification).
- Entities may be user-months (the aggregation of a user's activities over a month) or user-days. This is a simplification over what the PRODIGAL framework is capable of representing.
- The number of trials is the number of entities which were assigned a score.
- The number of positive trials is the number which involves, at least in part, inserted red team activities. (i.e., a user-day during which some red team actions were inserted together with the user's real actions is counted as a positive trial.)
- Methods that score only a subset of entities (e.g., only user-days where removable drives were employed) are measured against that subset of trials and the subset of positives contained within it. This is a method-centric measurement that must then be interpreted in the context of an overall system to determine how effective the method's contribution might be to the task of finding ITs.

Current metrics are the following:

- Receiver operator characteristic (ROC) curve is the well-known display of true positive rate vs. false positive rate .
- AUC is the area under the ROC curve. A random guess method results scores ~ 0.5. ROC and AUC show overall performance of a method at separating the positives from the negative trials.
- (Approximate) lift curve is the display of lift at each positive trial X rank of that trial. Lift(k) is defined as the ratio of observed precision of the method at rank k to the expected precision of a random guess, or (number of positives at or above rank k / k)/(number of positives/number of trials). The approximate curve is plotted just as the ranks where positives are found.
- Average lift is the average of the plotted points above. It measures the average workload reduction of an analyst who must "look" at all positives and relies on the method to help him triage his inputs.
- Number of positives at or above rank k (where k= 5, 10, 50, 100, 500 for user-month methods, and k=50, 100, 500, 1000, 5000 for user-day methods). These threshold metrics describe how well a method can support an analytic process with fixed bandwidth. If an analyst only has time to review 50 cases, then it is important to have good hits in the top 50, regardless of where the other positives occur in the ranking.

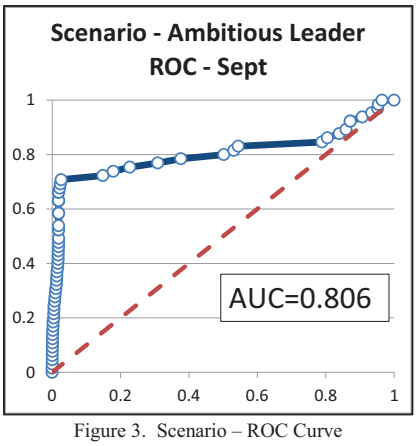

Figure 3. Scenario – ROC Curve

C. Details of Results

We ran 484 experiments on two months' of data with inserted red team scenario instances. September had 13 red team users with activity on 98 separate user-days and October, 6 users over 44 user-days. Experiments were run using the three methods for detecting starting points for further analysis; i.e., indicators, anomalies, and scenarios.

We describe an example of each type of starting point detector in detail along with results.

1) IP Thief – Ambitious Leader (Scenario Example): Figure 2 uses the anomaly detection language described in section IV to specify a system for targeting the Intellectual Property (IP) Thief Ambitious Leader scenario, cf. [3], in which we find a leader of a group of insiders who each steal a few files to be inconspicuous. To counter their strategy, we combine the file activity from the neighbors surrounding each user – known as an egonet – in the IM communication graph, making the leader more anomalous.

We start by filtering user-days to those with sufficient file activity $\left(C_{day}^{user}\right)$, then join those records with the IM user-neighbor adjacency list and sum up the features for each "neighbor" $\left(A_{day}^{neigh.}\right)$. We next add that total for each user to the user's own features and convert the feature totals into ratios ${}_1\left(A_{day}^{user}\right)$ that can be compared across egonets of different sizes, e.g. number of unique files to number of all files.

To limit the baseline population to users fitting the profile of a leader, we keep the users $(C_{mo.}^{user})$ with a high fraction of file accesses $(A_{mo.}^{user})$ fitting the manager role according to file extension $(C^{file-ext})$ and use this set as a simultaneous baseline to score $\left(S_{day}^{user}\right)$ each user-day.

As an additional indicator, we count ${}_2\left(A_{day}^{user}\right)$ phrases seen in IMs between users that fit the scenario (C^{phrase}) and finally combine $\left(\Lambda_{day}^{user}\right)$ with the anomaly scores.

Figure 3 shows the ROC curve resulting from testing this scenario on September data. Note that a significant subset of user-days rank very highly (one user ranks first, while five are identified in the top 100), while another set does so poorly as to be indistinguishable from random choice. In fact, the scenario completely ignores 33 user-days, since they do not fit with its assumptions. This is exactly what we would expect from a scenario specifically designed to identify individuals on days when they are

behaving like leaders of small groups exfiltrating IP.

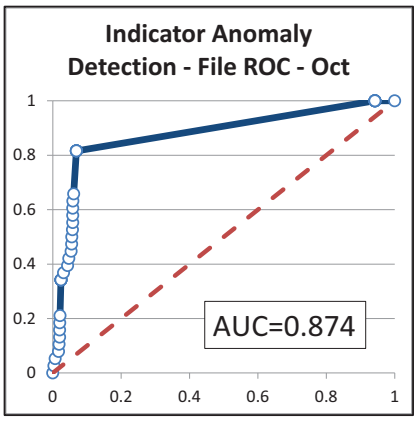

Figure 4. Indicator - ROC Curve

2) File Events Indicator (Indicator Example): In the File Events indicator method, we are looking for users who display abnormal behavior with respect to files, focusing on file events related to removable media drives and the number of distinct files that a user accesses. Component features include the ratio of file events on removable media to all file events and the ratio of distinct files on removable media to distinct files overall. Performance is also strong against the red team scenarios, which involved file copies to removable media. One advantage of this indicator is that it ignores user-days with no removable drive activity. Thus, when it works at all, it works reasonably well to focus other analytics. (See Figure 4 for the October ROC curve.) When it is not applicable, it can be safely ignored.

3) Relational Pseudo Anomaly Detection (RPAD) (Anomaly Detection Example): RPAD builds on the assumption that anomalies are rare to train a binary classifier, using the base population as negatives and an artificially generated set of points in the feature space as positives [1]. Extensive research into methods of feature normalization and construction of this pseudo-anomaly set have resulted in very high performance on our test data, AUC of 0.979 on the October data set. Figure 5 shows the ROC curve.

4) Repeated Impossible Discrimination Ensemble (RIDE) (Anomaly Detection Example): RIDE takes a novel approach of repeatedly training classifiers to distinguish between random partitions of the data [1]. Data points that are easily over-fit tend to be isolated anomalies. Features are derived from monthly aggregate-activity counts. Each user's aggregate is compared to other users' in the observed population and represented by the degree of statistical outlier-ness. This results in the best overall AUC of any anomaly detection algorithm, as well as placing 3 of the 6 target users in the top 100. (See Figure 6.)

5) Grid-based Fast Anomaly Discovery given Duplicates (GFADD) (Anomaly Detection Example): GFADD [8] applies a density estimation anomaly detector to nodes in a feature space. Nodes are judged anomalous with respect to their neighbors in the feature space. As a result, local anomalies in a complex organization may be detected, even if they are globally unremarkable. This is different from using a priori peer groups as base populations. We see from the ROC curve (Figure 7) that the algorithm appears to rank most red team targets, and indeed most nodes, as indistinguishable from normal. However, the top 100 nodes contain five red team targets, resulting in lift values in the 20-30 range. Furthermore, because the algorithm scores most other nodes very low, it "knows" when it is confident of its detection and so can be used effectively in a multi-method system. This characteristic of giving a few, high-confidence results only when the observed behavior is detectable is valuable in a multi-method system.

D. Overall Metrics

Table 3 shows results from indicator, algorithm, and scenario experiments run so far. AUC and Average Lift are as described earlier. The table has been sorted by AUC. Some methods operate over individual user-days, while others aggregate user behavior over the entire month before looking for anomalies. Columns labeled 5(0), 10(0), etc., represent counts of red team targets included in the top 5, 10, etc. for monthly methods, and the top 50, 100, etc., for daily methods.

All metrics are calculated with respect to the available ground truth (red team inserts). Note how poorly the URL

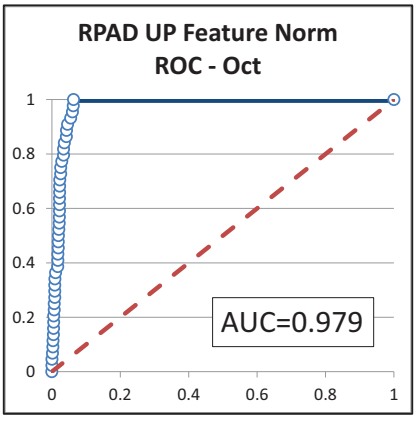

Figure 5. RPAD - ROC Curve

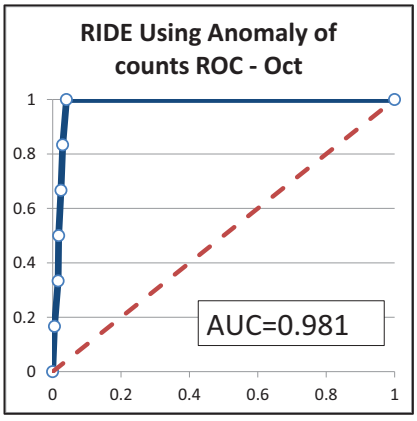

Figure 6. RIDE - ROC Curve

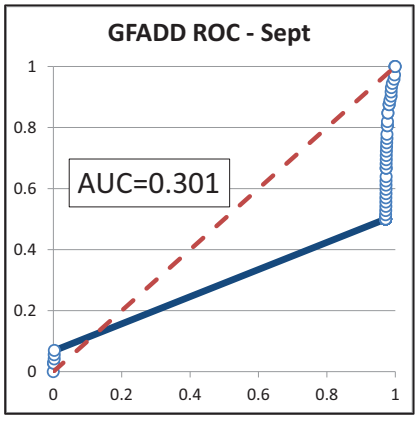

Figure 7. GFADD - ROC Curve

indicator (SAIC-7) performed. Since there are no significant web-based activities in the inserted scenarios, we would not expect it to score well. This points out a critical issue in testing unsupervised methods against an unknown set of targets – the metrics are only as perceptive as the ground truth they employ.

E. Effectiveness of Multiple Methods

While measures of performance, such as we have presented, are useful to determine how well particular methods are improving, they do not fully address contributions to the overall goal of the research – detecting known and unknown ITs. In particular, methods which produce the same rankings may be considered redundant, whereas methods which rank different targets highly may contribute to reducing misses in the overall system. To reduce redundancy, we would like different methods which work well under different conditions and on different data.

We have begun to measure the inter-method correlation of rankings. As an example, although some targets were missed by some methods in the October tests, all targets were ranked highly by at least one method. The two scenarios of inter-linked users each had at least one user who scored at the 99.5 percentile level (ranks 30 and 31) by

TABLE 3. OVERALL METRICS - EXPERIMENTS ON A VARIETY OF METHODS FOR DETECTING IT ANALYSIS STARTING POINT RUN IN THE PRODIGAL FRAMEWORK OVER 2 MONTHS' DATA. METRICS FROM SECTION V.B. ARE BASED ON INSERTED RED TEAM ACTIVITIES.

Month	Algo	Detection Method	5(0)	10(0)	50(0)	100(0)	500(0)	AUC	AvgLift
Oct	OSU-4	RIDE via unusualness of counts vs. company	0	0	1	3	6	0.981	26.18
Oct	UMASS-1	RPAD up feature normalization	2	2	5	11	37	0.979	30.33
Sept	OSU-3	Ensemble GMM Density Estimation, Raw Counts	0	0	0	0	12	0.970	26.17
Sept	UMASS-1	RPAD up feature normalization	2	2	3	11	72	0.970	17.42
Oct	OSU-3	Ensemble GMM Density Estimation, Raw Counts	0	0	0	0	6	0.970	15.84
Sept	OSU-1	GMM Density Estimation using Raw Counts	0	0	0	0	10	0.940	7.83
Sept	OSU-4	RIDE via unusualness of counts vs. company	0	0	0	2	10	0.920	8.05
Oct	UMASS-1	RPAD g129dm feature normalization	0	0	1	3	29	0.914	13.70
Oct	UMASS-1	RPAD raw feature set; naive bayes; uniform pseudo-anomaly	0	0	0	3	20	0.909	9.17
Oct	OSU-3	Ensemble GMM via unusualness of counts, vs company	0	0	0	0	2	0.906	5.32
Oct	OSU-1	GMM Density Estimation using Raw Counts	0	0	0	0	0	0.900	4.99
Oct	UMASS-2	RDE alpha version; raw feature set; 10k training	0	0	0	0	5	0.895	6.10
Sept	OSU-4	RIDE using Raw Counts	0	0	0	2	6	0.892	7.09
Oct	OSU-4	RIDE using Raw Counts	0	0	0	0	2	0.888	4.69
Oct	OSU-1	GMM Density Estimation via unusualness of counts, vs company	0	0	0	0	1	0.881	4.18
Sept	SAIC-6	Indicator Anomaly Detection - File	0	1	17	33	54	0.881	10.58
Sept	UMASS-1	RPAD raw feature set; naive bayes; uniform pseudo-anomaly	0	0	10	26	56	0.879	16.06
Oct	SAIC-6	Indicator Anomaly Detection - File	0	0	2	14	31	0.874	8.42
Sept	OSU-2	Cross Prediction via unusualness of counts, vs company	0	1	1	1	7	0.872	8.86
Sept	UMASS-2	RDE alpha version; raw feature set; 10k training	0	0	7	12	42	0.864	10.75
Sept	UMASS-1	RPAD dp feature normalization	2	4	20	26	57	0.863	24.07
Sept	SAIC-3	Scenario - IP Thief	0	0	7	16	54	0.851	9.79
Oct	GTRI-5	Temporal Based Anomaly Detection	0	0	0	0	12	0.849	6.14
Sept	SAIC-1	Max(Cross & Long Outliers)	0	0	0	1	14	0.846	3.99
Oct	SAIC-3	Scenario - IP Thief	0	0	0	3	15	0.839	7.34
Oct	OSU-2	Cross Prediction via unusualness of counts, vs company	0	0	0	0	1	0.833	3.15
Oct	SAIC-1	Max(Cross & Long Outliers)	0	0	0	0	0	0.828	3.27
Oct	SAIC-8	Indicator Anomaly Detection - File vs URL	0	0	2	8	28	0.824	6.02
Oct	SAIC-2	Scenario - Saboteur	0	0	0	0	15	0.810	3.07
Sept	SAIC-5	Scenario - Ambitious Leader	9	12	43	46	48	0.806	34.05
Oct	SAIC-5	Scenario - Ambitious Leader	6	7	12	12	15	0.789	80.20
Sept	OSU-3	Ensemble GMM via unusualness of counts, vs company	0	0	0	0	1	0.787	2.20
Sept	OSU-1	GMM Density Estimation via unusualness of counts, vs company	0	0	0	0	1	0.780	2.16
Sept	SAIC-2	Scenario - Saboteur	0	1	4	6	20	0.746	3.79
Sept	SAIC-8	Indicator Anomaly Detection - File vs URL	1	2	4	9	50	0.732	6.04
Oct	SAIC-4	Scenario - Fraudster	0	1	1	1	7	0.713	4.57
Oct	GTRI-4	Vector Space Models	0	0	1	2	2	0.694	8.64
Sept	SAIC-4	Scenario - Fraudster	0	0	0	1	10	0.693	1.62
Sept	GTRI-4	Vector Space Models	0	0	1	1	2	0.618	2.61
Sept	SAIC-9	Indicator Anomaly Detection - File vs URL vs Logon	0	0	3	4	8	0.530	1.26
Oct	SAIC-7	Indicator Anomaly Detection - URL	0	0	0	0	0	0.507	0.93
Sept	GTRI-5	Temporal Based Anomaly Detection	0	0	0	0	3	0.502	1.00
Sept	SAIC-7	Indicator Anomaly Detection - URL	0	0	0	1	5	0.477	0.91
Oct	CMU-6	Grid-based Anomaly Detection given Duplicates	1	1	1	2	3	0.465	1.77
Oct	SAIC-9	Indicator Anomaly Detection - File vs URL vs Logon	0	0	0	0	2	0.425	0.87
Oct	OSU-2	Cross Prediction using Raw Counts	0	0	0	0	0	0.388	0.92
Sept	CMU-6	Grid-based Anomaly Detection given Duplicates	2	5	5	5	5	0.301	2.19
Sept	OSU-2	Cross Prediction using Raw Counts	0	0	0	0	0	0.287	0.66

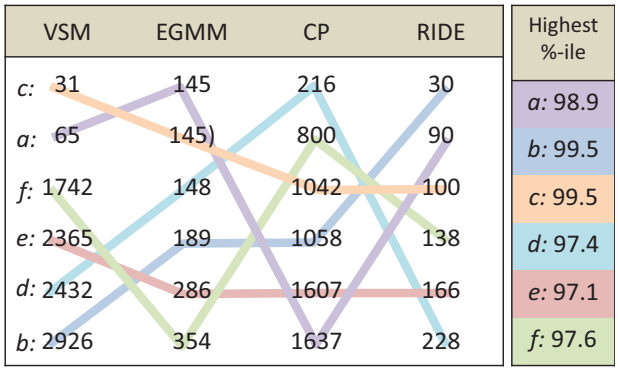

Figure 8. Target Users' Ranks by 4 AD Algorithms (October data)

different user-month methods. Considering these rankings as starting points for further analysis argues strongly for effectiveness of an overall system which can combine them.

Figure 8 shows the ranking performance of four monthly anomaly detection algorithms on the October data set. (There were six target users, a through f, in 5,500 total users for the month.) Lines connect each user's position across algorithms. Figure 8 also shows the highest rank (as a percentile) each user achieved in any algorithm. We note that each scenario had at least one representative at the 99.5 percent level (positions 30 and 31) from some anomaly detection algorithm. Subsequent analysis, based on observed user interactions should be able to find the other participants and uncover the joint scenario activity. Such analysis (more costly and involved) can be done from the top-scoring starting points.

VI. Summary and Next Steps

Domain-knowledge combined with structural anomaly detection is an effective way to triage data for IT analysis. Key considerations include normalization of features to reduce the effect of benign variations in activity levels; use of the meaningful base populations in anomaly detection, including both peer groups and linked sets of entities; and combination of methods which may individually be limited in scope.

Having established the effectiveness of the three methods for detecting starting points of IT analysis, we are ready to explore ways of combining their outputs. This will include additional domain semantics as the role of interpretation becomes more important in further reducing false positives.

We will also incorporate content derived from emails (i.e., topic and sentiment analysis) into our experiments. Anomaly-driven experiments would look for unusual patterns in the use of negative sentiment in words and phrases. Topic and sentiment analysis would fit into many of our scenario-driven experiments, and Table 4 lists specific examples.

Table 4. Examples of Semantic Content Experiments by Threat Scenario

Threat Scenario	Context Derived from Semantic Content
Saboteur; IP Thief-Entitled Individual	• Resentment • Detachment
IP Thief-Ambitious Leader	• Detachment • Influence within a group
Fraudster	• Personal distress
Rager	• Bursts of strong negative sentiment • Use of violent words/phrases

Acknowledgments

We thank the entire PRODIGAL team for their efforts in creating, implementing, and operating the experimental framework and algorithms. Funding was provided by the U.S. Army Research Office (ARO) and Defense Advanced Research Projects Agency (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] Unpublished works by David Jensen (University of Massachusetts) and Tom Dietterich (Oregon State University).

[2] R. Brackney, R. Anderson, *Understanding the Insider Threat: Proceedings of a March 2004 Workshop*. Santa Monica, CA: the RAND Corporation. 2004.

[3] 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.

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

[5] J. Hollywood, D. Snyder, K. McKay, J. Boon, Out of the ordinary: finding hidden threats by analyzing behavior. Santa Monica, CA: the RAND Corporation. 2004.

[6] 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.

[7] M. Keeney, et al., "Insider threat study: computer system sabotage in critical infrastructure sectors", United States Secret Service & the Software Engineering Institute, Carnegie Mellon University, May 2005.

[8] J-Y. Lee, U. Kang, D. Koutra, C. Faloutsos, "Fast anomaly discovery given duplicates," Carnegie-Mellon University, School of Computer Science, Dec. 2012, CMU-CS-12-146.

[9] M. Salem, S. Hershkop, S. Stolfo, "A Survey of Insider Attack Detection Research" in *Insider Attack and Cyber Security: Beyond the Hacker*, Springer, 2008.

[10] "SureView Proactive Endpoint Information Protection", Raytheon, February 13, 2013. Webpage: http://www.raytheon.com/capabilities/rtnwcm/groups/iis/documents/content/rtn_iis_sureview_datasheet.pdf

[11] J. Glasser, B. Lindauer, "Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data" Workshop on Research for Insider Threat, San Francisco CA, May 2013.