

# Detecting Insider Threats in a Real Corporate Database of Computer Usage Activity

Ted E. Senator,  
Henry G. Goldberg,  
Alex Memory,  
William T. Young, Brad Rees,  
Robert Pierce, Daniel Huang,  
Matthew Reardon  
SAIC  
{senatort, goldbergh, memoryac,  
youngwil, piercer, huangda, reardonmg}  
@saic.com

Jay-Yoon Lee, Danai Koutra,  
Christos Faloutsos  
Carnegie Mellon University  
{jaylee, danai, christos} @cs.cmu.edu

David A. Bader, Edmond Chow,  
Irfan Essa, Joshua Jones,  
Vinay Bettadapura,  
Duen Horng Chau,  
Oded Green, Oguz Kaya,  
Anita Zakrzewska, Erica Briscoe,  
Rudolph L. Mappus IV,  
Robert McColl, Lora Weiss  
Georgia Institute of Technology  
{bader,echow,irfan,jkj}  
@cc.gatech.edu  
{vinay, polo, ogreen, oguzkaya,  
azakrzewska3}@gatech.edu  
{Erica.Briscoe, Chip.Mappus,  
Robert.McColl, Lora.Weiss}  
@gtri.gatech.edu

Thomas G. Dietterich,  
Alan Fern, Weng-Keen Wong,  
Shubhomoy Das,  
Andrew Emmott, Jed Irvine  
Oregon State University  
{tgd, afern, wong, dassh, emmott, irvine}  
@eecs.oregonstate.edu

Daniel Corkill, Lisa Friedland,  
Amanda Gentzel, David Jensen  
University of Massachusetts  
{corkill, lfriedl, agentzel, jensen}  
@cs.umass.edu

## ABSTRACT

This paper reports on methods and results of an applied research project by a team consisting of SAIC and four universities to develop, integrate, and evaluate new approaches to detect the weak signals characteristic of insider threats on organizations' information systems. Our system combines structural and semantic information from a real corporate database of monitored activity on their users' computers to detect independently developed red team inserts of malicious insider activities. We have developed and applied multiple algorithms for anomaly detection based on suspected scenarios of malicious insider behavior, indicators of unusual activities, high-dimensional statistical patterns, temporal sequences, and normal graph evolution. Algorithms and representations for dynamic graph processing provide the ability to scale as needed for enterprise-level deployments on real-time data streams. We have also developed a visual language for specifying combinations of features, baselines, peer groups, time periods, and algorithms to detect anomalies suggestive of instances of insider threat behavior. We defined over 100 data features in seven categories based on approximately 5.5 million actions per day from approximately 5,500 users. We have achieved area under the ROC curve values of up to 0.979 and lift values of 65 on the top 50 user-days identified on two months of real data.

## Categories and Subject Descriptors

H.2.8 [Information Systems]: Database Applications – *data mining*.

## Keywords

Insider Threat, Anomaly Detection

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
KDD '13, August 11–14, 2013, Chicago, Illinois, USA.  
Copyright © 2013 ACM 978-1-4503-2174-7/13/08...\$15.00.

## 1. INTRODUCTION

Insider threat (IT) is a major problem for many organizations, including industry and the U.S. Government. [27] ITs may include intentionally malicious activities by authorized users, such as information system sabotage, intellectual property (IP) theft, fraud, and national security crimes (e.g., disclosure of classified information), as well as unintentional threats introduced inadvertently by careless use of computing resources. [3]

IT detection is more difficult than many other anomaly detection (AD) problems not only because insiders are knowledgeable about an organization's computer systems and procedures and authorized to use these systems, but also, and more important, because malicious activity by insiders is a small but critical portion of their overall activity on such systems. IT detection suffers from the technical challenges of very low signal-to-noise ratios (i.e., they are extremely rare but exceedingly important events) and dynamic threat scenarios (i.e., they are always changing because malicious insiders actively attempt to avoid being caught and also because computing and organizational environments evolve). IT instances consist of complex contextual combinations of activities, each of which may be authorized and legitimate when performed in different contexts or combinations. Often, much information necessary to fully discriminate between ITs and legitimate activities – by providing the context that explains apparently anomalous computer usage behavior – requires additional data sources such as employee personnel records, organizational charts, project assignments, work hours and locations. Furthermore, data about individuals are often protected from disclosure due to privacy needs.

To evaluate the feasibility of automated detection of ITs from computer usage data, Defense Advanced Research Project's Anomaly Detection at Multiple Scales (ADAMS) program [6] has collected a database of monitored computer usage activity in business organization, whose identity is not allowed to be disclosed publicly, of approximately 5,500 people. All data are used with permission in a closed testbed facility subject to all necessary privacy protections. Data are collected using a commercial tool called SureView® (Raytheon Oakley Systems, Inc.) [21], currently used by commercial and government

organizations to monitor suspicious individuals or specific targeted actions. SureView is resident on user workstations and captures user actions such as logins, file accesses, emails, instant messages, printer usage, browser usage (including URLs visited), process usage, etc. User identifications are anonymized by hashing. All collected data are treated as legitimate activity, a valid assumption given the rarity of malicious insider activity in the organization.

To provide realistic ground truth instances of ITs, an independent expert red team (RT) led by CERT develops scenarios of IT activity based upon case studies of known insider attacks [3][16] and augments the database with instances of such scenarios superposed on (sets of) users whose normal activity corresponds to the background characteristics of users involved in each scenario. The signal-to-noise ratio is approximately 0.2% of users and 0.08% of user-days. Scenarios are made available to researchers monthly, with “answer keys” consisting of identifiers of the artificially malicious users and descriptions of the scenario activities provided only after detection results have been generated. Each month’s data consists of approximately 1,000 actions per day per user, or about 5.5 million records per day.

## 2. APPROACH AND METHODS

A useful deployable IT detection system encompasses multiple data sources and multiple detection methods working together to identify and explain anomalies in context, with sufficient lift to permit human analyst review of all anomalies and action on all significant IT instances. This vision, based on how complex event detection systems operate in real environments, motivates and guides our work.

Our ADAMS project, called PRODIGAL (PROactive Detection of Insider threats with Graph Analysis and Learning), has developed, applied and evaluated multiple AD algorithms and supporting technologies based on models of different aspects of user behavior; over 100 semantic (i.e., domain-knowledge-based) and structural (graph-based) features; a schema representation for comparing results of different AD algorithms; a visual AD language; data extraction, loading, and transformation components; and an integrating software framework for experimentation. While not yet ready for deployment, we have achieved encouraging results that demonstrate the ability to detect the phenomena of interest in two months’ worth of real data.

### 2.1 Concepts and Architecture

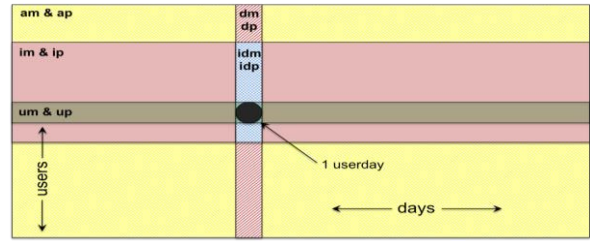
PRODIGAL supports regular data load and transform processes that enable feature computation, anomaly detection and results scoring from three types of starting points.

#### 2.1.1 Feature Construction

Feature definition combined domain knowledge and observable data knowledge. A retired operations officer from the U.S. intelligence community with expertise in how IT behaviors manifest in computer usage data identified three goals of

**Table 1: Feature Categories**

| 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<br>Ratio of URL uploads to URL downloads<br>Ratio of distinct removable drives to URL up/down loads |



**Figure 1: Feature Normalizations**

malicious insider activity related to information or systems: (1) destruction, (2) misuse or corruption, and (3) theft. Each goal has specific activities associated with five stages: (1) exploration, (2) experimentation, (3) exploitation, (4) execution, and (5) escape/evasion. These goals and stages cover ranges of behavior such as malicious insiders acting alone or in groups (with complicit and/or non-complicit group members) across time periods ranging from days to months. Corresponding observables in the SureView data were identified, and over 100 aggregate and ratio features in seven categories were deemed worthwhile to compute. (See Table 1.)

Relationship graphs from the base computer usage record were derived, including the email network among users, email addresses, computers, and messages; the printer network among users, printers, and computers; the web network among users, computers, domains, and individual URLs; the logon-logoff network between users and computers; and graphs composed of combinations of two or more of these networks.

We also explored a range of feature normalizations: raw values (**r**); median-difference and percentile of all users on all days in time period (**am & ap**); median-difference and percentile of all the user's days in time period (**um & up**); median-difference and percentile of all users on same day (**mp & dp**); median-difference and percentile of all users in group *i* on same day (**idm & idp**); and median-difference and percentile of all users in group *i*'s days in time period (**mp & ip**). Intuitively, for a given feature: normalizing each day's value to all the user's days should highlight unusual days; normalizing each day's value to all users on that same day should dampen day-to-day, across-the-board variations; normalizing each day's value to a user's group should heighten variation from peers. Unlike median-difference normalization, percentile normalization is insensitive to distant outliers. The relationship between these normalizations is illustrated in Figure 1. We found that percentile normalization by each user's days (**up**) gave the best results on our collected data.

#### 2.1.2 Anomaly Schema

*Representation*  
We designed and implemented a common data structure to capture descriptions and results of every AD experiment. Table 2 depicts this data structure. For each run, we allow algorithms to compute a raw score, an ordered ranking of all entities, a normalized

**Table 2: Anomaly Detection Data Structure**

| ResultScore         | ResultMetadata       |
|---------------------|----------------------|
| runID (KEY)         | runID                |
| flowID (KEY)        | flowID               |
| algoID (KEY)        | algoID               |
| dataTypeId (KEY)    | dataTypeId (KEY)     |
| nodeID              | entityXtent (KEY)    |
| rawScore [optional] | featureID [optional] |
| normScore           | EntityTemp           |
| rankedScore         | popXtent             |
| Rank                | popSubXtent          |
| endDate             | popTemp              |
| analystScore        | scoreMean            |
| hasAnalystScore     | scoreStdev           |
| analystUpdateDate   | scoreCount           |
|                     | parameters           |

**Table 3: Threat Scenarios**

| Name   | Distinguishing Indicators/Anomalies  |
|--|--|
| Saboteur   | <ul style="list-style-type: none"> <li>Indicators: URL; File; Logon</li> <li>Anomalies: File accesses in relation to peer group in LDAP</li> </ul>   |
| Intellectual Property (IP) Thief-Ambitious Leader    | <ul style="list-style-type: none"> <li>Indicators: URL; File; Printer; Login; Email</li> <li>Anomalies: File accesses and email communication graph in relation to peer group (manager) in LDAP</li> </ul> |
| Intellectual Property (IP) Thief-Entitled Individual | <ul style="list-style-type: none"> <li>Indicators: URL; File; Printer; Login; Email</li> <li>Anomalies: File accesses compared to peer group (technical) in LDAP</li> </ul>                                |
| Fraudster  | <ul style="list-style-type: none"> <li>Indicators: Login; processes; files; and URL</li> <li>Anomalies: Email and URL compared to groups (non-technical)</li> </ul>  |
| Careless User  | <ul style="list-style-type: none"> <li>Indicators: File, Email, URL, Process</li> <li>Anomalies: Processes run compared to user and LDAP (function) group</li> </ul>                                       |
| Rager  | <ul style="list-style-type: none"> <li>Indicators: Email, IM, Login</li> <li>Anomalies: sentiment and topics in emails sent</li> </ul>   |

score, and percentile (ranked) scores. The raw scores may be on different scales and difficult to combine across algorithms. Normalized scores and ranks enable us to compare scores across algorithms. Distributional information such as mean and standard deviation allows us to determine degrees of anomaly.

### 2.1.3 Starting Points

PRODIGAL’s AD methods serve as the first stage in a multi-layered detection process [26], leading to further data acquisition and analysis. Three kinds of starting points, all of which use some sort of domain knowledge, are employed.

- Indicators – single observables or counts, known or suspected to correlate with IT activities, but not necessarily tied to specific IT scenarios
- Anomalies – unusual patterns, typically high-dimensional, resulting from AD algorithms working over vectors of features, graphs of relationships, or sequences of activities
- Scenarios – matches of specifically designed patterns that correspond to known or suspected computer usage activities.

We constructed several scenarios from documented patterns of known malicious insider behavior. Detectors for these patterns were implemented in the PRODIGAL framework using available features, indicators, and outlier detection algorithms, as well as peer groups discovered in the cyber activities by graph-based community detection algorithms. A detailed example is found later in this paper. Table 3 lists the scenarios with the indicators and anomalies utilized by each.

## 2.2 Novel Algorithms

Multiple AD algorithms based on suspected scenarios of malicious insider behavior, indicators of unusual activities, high-dimensional statistical patterns, temporal sequences, and normal graph evolution were developed and evaluated. Algorithms and representations for dynamic graph processing provide the ability to scale as needed for enterprise-level deployments on real-time data streams. Each of the algorithms is based on the idea that AD consists of comparing the observed data to data that would result from entirely “normal” behavior. The algorithms differ not only according to what aspects of normal behavior they model, but also to the techniques they use to determine if observations differ from normal and the form of their output. In addition to the AD algorithms, we include descriptions of novel supporting

algorithms that compute relevant peer groups such as those that perform community detection based on the graph of user interactions and others that provide the foundation necessary for scaling-up to a real-time deployed system in an organization of several 100k people.

### 2.2.1 Relational Pseudo-Anomaly Detection (RPAD)

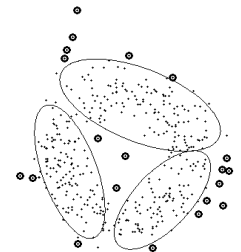
RPAD learns a model of normal behavior by first taking a sample of the observed data instances, treating them as non-anomalous, and constructing an equal number of *pseudo-anomalies*. Pseudo-anomalies are drawn from a joint distribution in which every feature is drawn independently from its marginal distribution. RPAD constructs a classifier to distinguish the observed data instances from the pseudo-anomalies. When given a new instance, RPAD combines the classifier’s prediction with the pseudo-anomaly distribution to determine whether the instance is anomalous. This approach produces a representation of the joint distribution that is sufficient for anomaly detection and that is highly efficient to learn and apply [13][14]. RPAD learns the classifier and assesses all entities in less than 10 minutes on a data set of 131,729 entities with 83 features, and it outputs a score corresponding to the degree of anomalousness of an entity extent.

### 2.2.2 Relational Density Estimation (RDE)

RDE is a simple estimator of joint probability that assumes feature independence. Each marginal distribution is modeled using a kernel density estimator, and the joint probability is assumed to be a simple product of these marginal distributions. Anomalous points are those that have low probability estimates under this joint distribution. The probability estimates themselves are biased when the independence assumptions are violated, although rankings can still be accurate despite these biases. As with RPAD, the resulting estimator of the joint distribution is efficient to construct and apply, although its performance is generally below that of RPAD.

### 2.2.3 Gaussian Mixture Model (GMM)

A GMM models the density of the data in feature space using a weighted sum of Gaussians. A data point is ranked as an anomaly based on its modeled density (lower density meaning more anomalous). For instance in Figure 2, data are input to the algorithm as a set of feature vectors (shown as points). The model is trained using the EM algorithm [7], and the fitted Gaussians are shown as ellipses. The 20 lowest-density points are shown in black. To speed up training, we implemented an incremental algorithm [20]. The number of Gaussians  $k$  is chosen via cross-validation.



**Figure 2: GMM with 3 components.**

Figure 2 illustrates that a single GMM can have gaps between the Gaussians, and it will underestimate the density in those regions. We address this by training an ensemble of GMMs. Ensembling is a general tool for improving learning methods [8, 29]. Each GMM is trained on a bootstrap replicate of the original data, and we also vary the number of components  $k$ , which eliminates the need to choose a specific  $k$ . EGMM discards individual GMMs that do not achieve a minimum likelihood. The fitted density is the simple average of the densities of the individual models. The user must

### 2.2.4 Ensemble Gaussian Mixture Model (EGMM)

Figure 2 illustrates that a single GMM can have gaps between the Gaussians, and it will underestimate the density in those regions. We address this by training an ensemble of GMMs. Ensembling is a general tool for improving learning methods [8, 29]. Each GMM is trained on a bootstrap replicate of the original data, and we also vary the number of components  $k$ , which eliminates the need to choose a specific  $k$ . EGMM discards individual GMMs that do not achieve a minimum likelihood. The fitted density is the simple average of the densities of the individual models. The user must

specify the number of models to fit, the range of  $k$  values to try, and the minimum likelihood threshold. The results are relatively insensitive to these choices [5].

### 2.2.5 Repeated Impossible Discrimination Ensemble (RIDE)

Suppose that we randomly partition the data into two equal sets ( $A$  and  $B$ ) and then apply a supervised learning algorithm to attempt to discriminate  $A$  from  $B$ . By construction,  $A$  and  $B$  have the same probability distribution, so it is impossible to tell them apart. However, flexible machine-learning algorithms can be encouraged to overfit the data and find ways of discriminating some points in  $A$  from  $B$ . The key claim of RIDE is that outlier points are easier to discriminate *at random* from other points [5]. Hence, if—across many random  $A/B$  splits—a point  $x$  is persistently overfit, then that point is ranked as an anomaly. Specifically, we employ flexible logistic regression (boosted regression trees with a logistic link function [22]) to discriminate  $A$  from  $B$ . If a point  $x$  has fitted probability  $p$  of belonging to class  $A$  (versus  $B$ ), it is assigned an anomaly score  $2|p-\mu|$ , where  $\mu$  is the median fitted probability of belonging to class  $A$ . We perform 100 random  $A/B$  splits and average the anomaly scores to obtain the final ranking.

### 2.2.6 Cross Prediction

Suppose feature vectors have length  $J$ . Cross prediction [5] learns  $J$  conditional probability models of the form  $P(x_j|x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_J)$  by applying a supervised learning algorithm (a modified version of Quantile Forests [18]) denoted as  $P(x_j|x_{-j})$ . It then scores a point  $x$  as anomalous according to the score  $1-\prod_j P(x_j|x_{-j})$ . The underlying assumption is that there are correlations among the features so that the value of one feature can be predicted from the others. If this assumption holds, then a point is anomalous if one or more of its features cannot be predicted from the others (so  $P(x_j|x_{-j})$  is small). The product of the  $P(x_j|x_{-j})$  values is very small if many of the feature values are hard to predict, so this gives a high anomaly score to such points. We employ a cross-validation procedure to decide which features  $j$  to include in the product and delete those features that are not predictable.

### 2.2.7 Grid-based Fast Anomaly Detection Given Duplicates (GFADD)

Many traditional outlier detection methods are slow due to large numbers of duplicate points. Given a cloud of multi-dimensional points, GFADD [15] detects outliers in a scalable way by taking care of the major problem of duplicate points. Fast Anomaly Detection given Duplicates (FADD) solves duplicate problems by treating them as one super node rather than considering them separately. Moreover, GFADD applies a  $k$ -dimensional grid on

the  $k$ -dimensional cloud of points and treats as super nodes only the grid cells that consist of more points than the number of nearest neighbors we are interested in. This method achieves near-linear runtime given duplicates, while Local Outlier Factor (LOF), the traditional outlier method that constitutes our baseline, has quadratic runtime. GFADD can spot anomalies in data sets with more than 10 million data points, while the traditional LOF algorithm runs out of memory even for 20,000 data points.

One example of AD using GFADD in PRODIGAL is provided in Figure 3. On the two-dimensional space of log (net event count) vs. log (remote event count), GFADD spotted circled points where frequency of a remote event is much higher than that of net events.

### 2.2.8 Vector Space Models

VSM [2] deals with sequential data that represents, e.g. a user’s behavior over time. Specifically, VSM operates over an input dataset consisting of a set of (finite-length) sequences of events, where the events are discrete symbols drawn from a known, finite alphabet. In general, a raw dataset must be preprocessed to segment and abstract data appropriately into an amenable format. Given a set of input sequences, we create a single bag-of-words style feature vector for each sequence through standard n-gramming. This feature vector consists of a count for each possible length= $n$  sequence given the finite alphabet. The feature vector for a given sequence represents the number of times each n-gram appears in that sequence. We then compute pairwise cosine similarity over these feature vectors and assign each sequence a score based on its proximity to its nearest  $k$  neighbors according to this cosine similarity measure. We next rank all sequences based on this computed score. The higher a sequence’s rank, the more anomalous it is believed to be based on this algorithm. Recent approaches that use variants of the n-gram approach to represent activities in terms of their local event subsequences [12] have inspired this approach.

### 2.2.9 Temporal-Based Anomaly Detection

Temporal-based AD aims to track and model user behavior at different temporal scales for detecting anomalies [17]. The hypothesis is that anomalous behavior is more easily and accurately recognized using multiple observations at different temporal resolutions. Further, users’ ability to mask anomalous behavior is diminished when observed at multiple temporal resolutions. We have applied temporal-based methods using hidden Markov models (HMMs) and particle filters for tracking users. Extending traditional particle filters improves their sensitivity. We use a gradient method to optimize particle cloud samples to more tightly track observations. By tightening the particle cloud to the observations while minimizing the effect to the variance of the particle cloud, the filter becomes more sensitive to anomalies.

### 2.2.10 STINGER

STINGER is a dynamic graph data structure capable of maintaining and representing temporal and semantic graphs with millions to billions of relationships between entities [1]. Edges and vertices are given types, weights, timestamps, and physical identifiers. Adjacencies are stored in semi-dense lists of blocks of edges. The data structure’s basic operations are thread-safe, which enables the insertion and deletion of edges and vertices at rates of millions of updates per second on a modern multicore shared-memory x86 platform [9]. Combining this with a convenient parallel filtering and traversal allows algorithms to easily process and understand the structural changes in the graph over time to

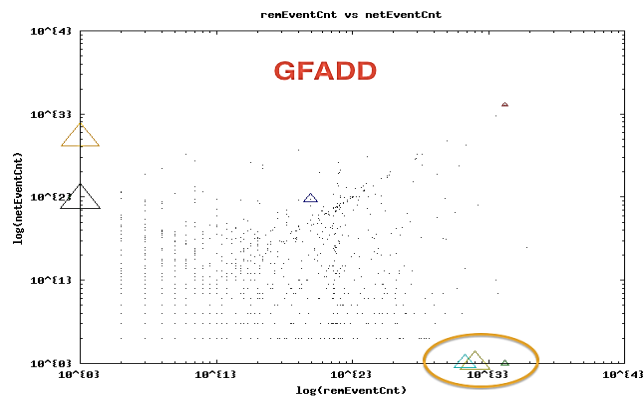


Figure 3: Anomaly detection using GFADD.

identify anomalous relationships, actors, and groups. In PRODIGAL, STINGER serves as a platform to enable dynamic graph analysis techniques on the relationships inherent in the dataset. Graph analysis speed enables recomputation of relational context, not only with new inputs, but more important, to permit preliminary detection results to refocus further analysis.

### 2.2.11 Community Detection

Our community detection algorithm identifies community structures within graphs stored in STINGER based on communication patterns, social structure, and resource sharing. The algorithm takes a similar approach to [25] and leverages STINGER’s parallel insertion and deletion for in-place operation. All vertices are initialized to their own communities, and all edges are scored. Trees of edges within the graph are constructed based on high-scoring edges and paths. These trees are contracted into the root communities, and the process is repeated for the resulting community graph until convergence of the edge scores. The algorithm is agnostic to the types of edges and vertices in the graph and the scoring function used; however, the scoring function can be defined to handle types separately. The algorithm can be applied repeatedly using different scoring functions to seek out certain traits in the graph. Each iteration of the algorithm finds groups at a coarser granularity. Community members that connect between communities can be explored to find shifting allegiances. [23]. The resulting communities can serve as baseline comparison groups, and individual communities can be extracted.

### 2.2.12 Streaming Community Detection

When dealing with massive and rapidly evolving data, static community detection approaches may not keep up with the change in data because they require a very costly total recalculation after any changes. Streaming community detection computes incremental updates to the detected communities when data changes. A community graph is maintained over time. We assume that the effect of most data changes will be local. When updates do occur, we break up the maintained community graph by removing a set of vertices affected by the changes from their communities. The resulting, broken-down community graph is then reclustered by a static community detection algorithm. The speed-up results because the broken-down community graph is much smaller than the original graph, enabling the graph community structure to be tracked over time and significant changes to be detected. Additional details can be found in [23].

### 2.2.13 Seed Set Expansion

Given a graph and a set of seed vertices in the graph as input, seed set expansion produces a subgraph, or community, that best contains the seed vertices. When entities are represented as vertices and their interactions as edges in a graph, given several entities of interest, perhaps flagged as anomalous, we may wish to know other entities related to these flagged ones. It may also be useful to extract a relevant subgraph containing these flagged entities in order to perform more computationally intensive analysis. We use a greedy, modularity maximizing approach to expand the seed set into a relevant subgraph. Details of the approach can be found in [24] with additional work in progress. The initial subgraph contains the seed vertices. In each iteration, the change in modularity that would occur by adding vertices to the subgraph is calculated and the highest scoring vertices are added.

### 2.2.14 Betweenness Centrality for Streaming Graphs

Within a graph, centrality metrics score the relative importance of each vertex and/or edge in the graph. In the case of betweenness

centrality, the score given to each vertex  $s$  is the sum of the fraction of the shortest paths between vertices  $r$  and  $t$  that  $s$  lies on for all pairs  $r$  and  $t$  ( $r \neq s \neq t$ ). For PRODIGAL, we have developed a novel algorithm for computing betweenness centrality for streaming graphs [11]. The new algorithm avoids computing betweenness centrality scores for vertices that do not have new paths going through them following the insertion of a new edge. It uses restricted breadth-first traversals to identify vertices whose shortest paths were affected and vertices whose betweenness centrality scores are affected by the those shortest path count changes. This significantly reduces the number of vertices and edges that are traversed and improves update response time so that approximations to betweenness can be maintained at the rate of change of the graph. We have shown in [11] that the algorithm is up to two orders of magnitude faster than recomputing. In PRODIGAL, this algorithm can be used to search out key actors and information flow points to find weaknesses and leaks within the organization. In [10] we extend our work and show how to increase the parallel scaling for computing betweenness centrality for streaming and static graphs. This approach also improves performance on a single processor.

### 2.2.15 Interactive Graph Exploration (Apolo)

Apolo [4] is an interactive graph visualization component of PRODIGAL, which helps analysts understand the relationships among anomalous entities flagged by various PRODIGAL algorithms. Apolo supports real-time graph exploration over million-node graphs, such as querying for nodes by their attributes, grouping nodes into super nodes to reduce cluster and promote understanding, visualizing subgraphs and interactively expanding them by bringing in neighboring nodes and edges. Unlike most graph visualization tools that keep the full graph in the main memory, which prevent them from handling large datasets, Apolo keeps the graph in an embedded database (SQLite), significantly reducing memory needs while maintaining high speed and scalability. Apolo also provides a built-in machine learning algorithm (Belief Propagation) that helps analysts find the most relevant subgraphs to visualize, given the analyst’s current nodes of interest. The algorithms works by computing a proximity score for each node based on how far it is from those nodes of interest (e.g., number of hops away, and the number of paths leading to them). Nodes with higher proximity scores are deemed more relevant and will be displayed to the analyst.

## 2.3 Anomaly Detection Language

Effective AD requires combining multiple methods applied to different baseline and peer group populations over distinct time

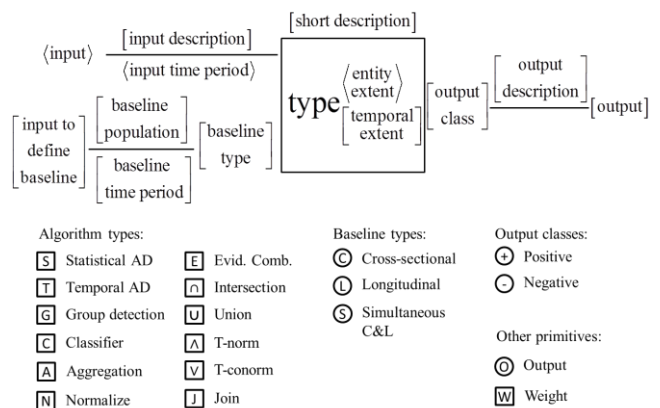


Figure 4: Anomaly Detection Language Syntax

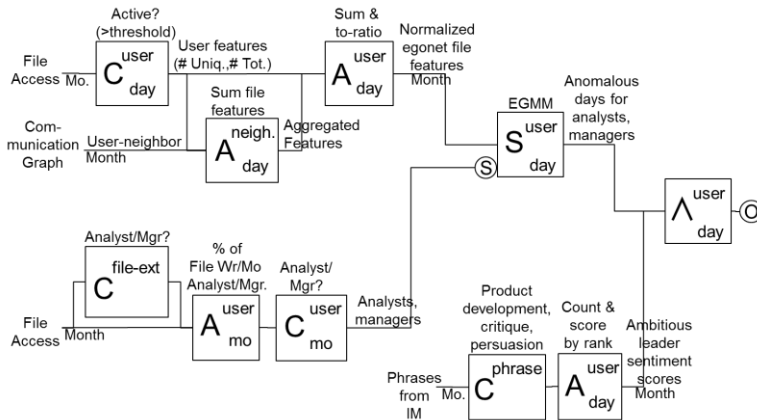


Figure 5: Language Example, IP Thief Ambitious Leader

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 AD 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 4; we refer to [19] for details concerning the syntax, and instead illustrate the language with an example, shown in Figure 5, targeting the IP Thief/Ambitious Leader scenario. In this scenario, an individual enlists others to steal portions of IP that, when combined, enable them to start a competing organization. We begin by filtering user-days to those with sufficient file activity, join those with the IM user neighbor adjacency list, and sum up the features for each neighbor. We next add that total for each user to the user’s own features and convert the feature totals into ratios that can be compared across egonets of different sizes. To limit the baseline population to users fitting the leader profile, we keep those with a high percentage of file accesses in that category and use this set to score each user-day. As an additional indicator, we count phrases seen in IMs between users that fit the scenario, and we finally combine with the anomaly scores.

### 3. EXPERIMENTS AND RESULTS

PRODIGAL Framework explored combinations of features, entities, baselines and peer groups, and detection methods for ITs. [28] We used two separate months of data, with an unknown in advance number of independent red team inserts in each. We ran a wide variety of indicator detectors, AD algorithms, and scenario-based detectors. We applied several performance metrics to understand the effectiveness of each, as well as of the whole suite of methods, for IT detection.

#### 3.1 Test Data

For the experiments discussed in this paper, the primary red team scenario consisted of three insiders who collude over instant messages and corporate email to steal IP and form a new

company. This is similar to the IP Thief Ambitious Leader scenario discussed in section 2.3, without the presence of an identifiable leader. The inserted scenario was unknown to the detectors (and to the research team) prior to the experiments. We discovered that the red team had inserted two variants of this scenario over the months of September and October 2012, inserting a total of six instances. A second scenario simulated users’ circumventing SureView’s data collection.

#### 3.2 Metrics

The experiments reported here measure AD performance on entity-extents, which are either user-months (the aggregation of a user’s activities over a month) or user-days. Metrics were chosen not only to measure detection accuracy of the individual algorithms, but also to measure their contribution to the overall task of providing leads to an analyst. We computed receiver operator characteristic (ROC) curves and area under the curve (AUC) as well as the approximate lift curves and Average Lift. AUC directly estimates the probability that a randomly chosen positive entity extent will be ranked higher than a randomly chosen negative one. Average lift estimates the improvement in target density delivered to later stages of a multi-stage detection process. To directly measure impact on an overall detection system, we report numbers of targets in the top k. 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) allow us to estimate the feasibility of employing a method of detection to provide starting points for analysis which may only be able to input a fixed number of such leads.

#### 3.3 Results

We ran 484 experiments on two months of data with inserted RT scenario instances. September had 13 RT users with activity on 98 separate user-days and October had 6 RT users with activity on 44 user-days. Examples and discussion of results from five detection methods are followed by a summary of the entire set of results.

##### 3.3.1 IP Thief Ambitious Leader Scenario Detector

We constructed a scenario-based detector that corresponds to the design in section 2.3 for the IP Thief Ambitious Leader scenario. Figure 6 shows the ROC curve resulting from testing this scenario detector on September data. Note that a significant subset of user-days rank very highly – one user ranks first, and five are identified in the top 100, while another set does so poorly as to be indistinguishable from random choice. In fact, the detector completely ignores 33 user-days, because 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 behave like leaders of small groups exfiltrating IP.

##### 3.3.2 File Events Indicator

With 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.

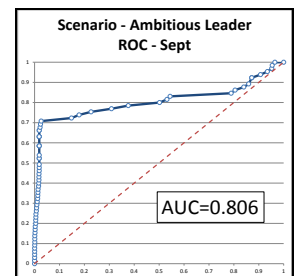


Figure 6: Scenario - ROC Curve

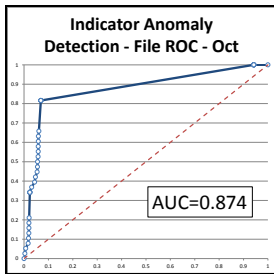


Figure 7: Indicator - ROC Curve

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 RT 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 well enough to enable subsequent analytics. (See Figure 7 for the October ROC curve.) When it is not applicable, it can be safely ignored.

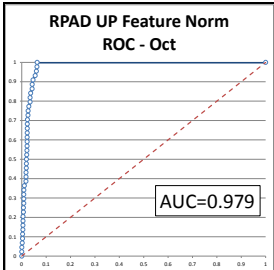


Figure 8: RPAD - ROC Curve

### 3.3.3 Relational Pseudo Anomaly Detection

Results from RPAD were highly consistent across both test months. Feature normalization resulted in very high performance on our test data, achieving an AUC of 0.979 on the October data set. Figure 8 shows the ROC curve.

### 3.3.4 Repeated Impossible Discrimination Ensemble

RIDE was also consistent across the test months. Features were 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 anomaly. This resulted in the best overall AUC of any AD algorithm, as well as placing all six target users in October in the top 5%. The lift curve is shown in Figure 9.

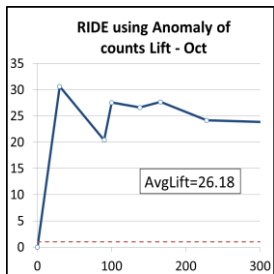


Figure 9: RIDE - Lift Curve

### 3.3.5 Grid-based Fast Anomaly Discovery given Duplicates

GFADD estimates the degree of anomaly of nodes in a graph with respect to their neighbors in the graph. 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. The lift curve (Figure 10) shows the top 100 nodes contain five RT targets, resulting in lift values in the x20-x30 range. Furthermore, because the algorithm only returns high-confidence results, it provides reliable starting points.

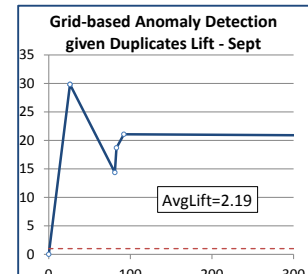


Figure 10: GFADD - Lift Curve

### 3.3.6 Overall Metrics

Figure 11 shows results from multiple indicator, algorithm, and scenario-based experiments. The table has been sorted by the column labeled 500(0), which records the number of positive hits in the top ranked 500 user-months/top 5,000 user-days (for algorithms which focus on users' activities over a month/day, respectively). AUC and average lift are as described earlier. Note that methods' performance per AUC or AvgLift was not always reflected in their raw hit output. For example, Ambitious Leader Scenario on September data hit 9 of 98 targets in the top 50, and 48 in the top 5,000. Shading is proportional to the range of values over each metric separately (hits, AUC, and AvgLift). All metrics are calculated with respect to the available ground truth (i.e., RT inserts). As expected, the URL indicator detector performed poorly, due to the lack of significant web-based activity in the inserted scenarios. This points out a critical issue in

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

Figure 11: Overall Metrics

|    | VSM  | EGMM | CP   | RIDE | Highest %-ile |
|----|------|------|------|------|---------------|
| c: | 31   | 145  | 216  | 30   | a: 98.9       |
| a: | 65   | 145  | 800  | 90   | b: 99.5       |
| f: | 1742 | 148  | 1042 | 100  | c: 99.5       |
| e: | 2365 | 189  | 1058 | 138  | d: 97.4       |
| d: | 2432 | 286  | 1607 | 166  | e: 97.1       |
| b: | 2926 | 354  | 1637 | 228  | f: 97.6       |

Figure 12: Target Users' Ranks by 4 AD Algorithms (October) data)

testing unsupervised methods against an unknown set of targets – the metrics are only as perceptive as the ground truth they employ. Two critical points to take from this are that experiments must be repeated with different inserted targets to explore the effectiveness and robustness of methods, and ways of combining multiple methods must be developed so that the system can build upon the strengths of each without interfering with one another.

### 3.4 Effectiveness of Multiple Methods

Measures of performance of individual algorithms are useful; however, we are also interested in how combining these algorithms contributes to overall system effectiveness. Methods which produce the same rankings may be redundant, whereas methods which rank different targets highly may contribute to reducing misses in the overall system. To meaningfully measure this inter-method correlation, it is important to consider the correlation of non-targets as well as targets.

Figure 12 shows the ranking performance of four Monthly AD algorithms on the October data set. Lines connect each of the six RT augmented users' positions across four algorithms. Figure 12 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% level (positions 30 and 31) from some AD algorithm. Subsequent analysis, based on observed user interactions, likely would find the other participants and uncover the joint scenario activity. Such analysis, although more costly and involved, can be initiated from the top-scoring starting points.

### 3.5 Feature Strength Estimation

Eighty-three scalar features are obtained from the collected data. An interesting question is: *Which of these features are the most useful in detecting anomalies?* Intuitively, the potential usefulness of a percentile-normalized feature when used in combination with other features depends on the distribution of its values: features whose values have high variability (“diversity”) have more potential unusual combinations with other features—a greater “feature strength”—than features whose values are nearly constant.

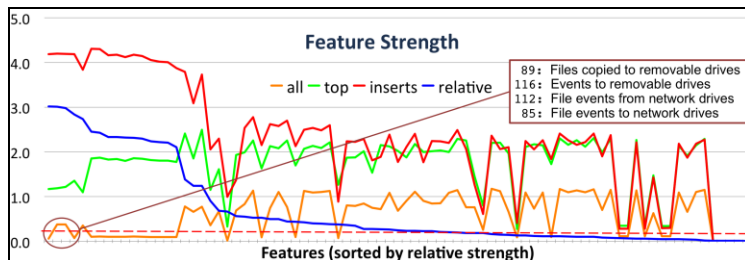


Figure 14: Computing Relative Feature Strength

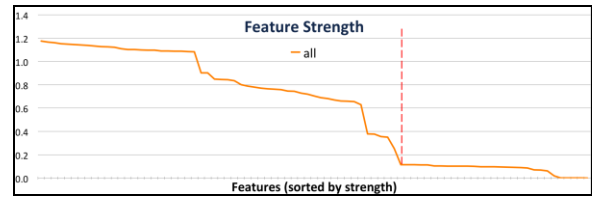


Figure 13: Feature Strength

Figure 13 shows the feature strength (average squared distance from the mean) of 83 features over a month of **all** user-days (features on x-axis sorted by strengths). The dotted vertical line gives a naïve “cutoff” for weak features.

Figure 14 extends this analysis to address the following questions: *What features contribute most to a high pseudo-anomaly detection score?* **Top** shows feature strengths computed using only the top 5,000 RPAD-ranked user day instances. *What features contribute most to high insert scores?* **Inserts** shows feature strengths computed using only the RT inserts. *What features differentiate inserts from high non-insert scores?* **Relative** shows *relative* feature strengths computed as the distance between the **top** 5,000 feature strengths and the **inserts** feature strengths. In Figure 14, the x-axis feature ordering is by relative strength, and the weak feature cutoff line is shown horizontally. Features with an all-instances strength (**all**) below the cutoff line would be excluded by a weak feature cutoff strategy.

We found that feature strength analysis indicates that low all-instances strength features were significant when used jointly and that a naïve strategy that focused only on the strongest all-instances features would miss these important feature correlations. We validated this observation by running RPAD without the 29 all-instances-strength features below the cutoff line. The AUC dropped from 0.970 with all features to 0.793 without those “weak” all-instances-strength features.

The strongest relative features, determined using only RT insert user-day labeling and without any semantic knowledge of RT scenarios, were consistent with user behaviors in the scenarios once those were revealed to us. (The top four relative-strength features are shown in the callout box in Figure 14.)

We concluded that manual methods, in which analysts focus on small numbers of apparently high-value features, are unlikely to detect anomalies characteristic of IT scenarios, because they would miss the most important combinations of individually weak features and because analysts would be overwhelmed by the number of relevant features.

## 4. CONCLUSIONS AND FUTURE WORK

The work reported here demonstrates the feasibility of detecting the weak signals characteristic of ITs using a novel set of algorithms and methods. However, additional research and engineering is needed to enable these techniques to be useful for real analysts in an integrated system. Some of the methods that we

have developed may be useful as components of existing systems; and we are actively exploring these possibilities with various government users. We also anticipate live field testing of these methods in the near future.

In the near term, we are continuing to investigate how to improve our coverage of the IT space by combining results from indicators, anomalies, and scenario-based detectors and by exploring different combinations of algorithms. We will be extending our feature coverage to include topic and sentiment detected in the contents of email and IM



communications. And we are continuing our experiments each month with additional data and new scenarios provided by the red team. These ongoing experiments enable us to further validate our current conclusions and to cover a wider range of IT scenarios.

In the longer term, we hope to be able to use additional data sources to provide more accurate detection and also to generate explanations of anomalous behavior that can be understood by human analysts. We believe that this is a key to developing a system that is useful by analysts instead of computer scientists. We also need to scale-up to continuous operation on organizations of about 100,000 users.

## 5. ACKNOWLEDGMENTS

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. This document is Approved for Public Release, Distribution Unlimited.

## 6. REFERENCES

- [1] Bader, D.A., et al. 2009. STINGER: Spatio-Temporal Interaction Networks and Graphs (STING) Extensible Representation. Technical Report, May 8, 2009.
- [2] Bettadapura, V., et al. 2013. Augmenting Bag-of-Words: Data-Driven Discovery of Temporal and Structural Information for Activity Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
- [3] Capelli, D., Moore, A., and Trzeciak, R. 2012. *The CERT Guide to Insider Threats*. Addison Wesley, Boston, MA
- [4] Chau, D. H., et al. 2011. Apolo: making sense of large network data by combining rich user interaction and machine learning. In CHI 2011.
- [5] Das, S., Emmott, A., Dietterich, T. G., Wong, W-K, Fern, A. 2013. Anomaly Detection with Ensembles of Statistical Models. Technical Report, Oregon State University.
- [6] Defense Advanced Research Projects Agency 2010. Anomaly Detection at Multiple Scales (ADAMS) Broad Agency Announcement DARPA-BAA-11-04. Arlington VA.
- [7] Dempster, A. P., Laird, N. M., and Rubin, D. B. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1-38.
- [8] Dietterich, T. G. 2000. Ensemble Methods in Machine Learning. In J. Kittler and F. Roli (Ed.) *First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science* (pp. 1-15). New York: Springer Verlag.
- [9] Ediger, D., et al. 2012. STINGER: High Performance Data Structure for Streaming Graphs. IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, September 10-12, 2012.
- [10] Green, O., Bader, D.A. 2013. Faster Betweenness Centrality Based on Data Structure Experimentation. 13<sup>th</sup> International Conference on Computational Science (ICCS).
- [11] Green, O., McColl, R., and Bader, D.A. 2012. A Fast Algorithm for Streaming Betweenness Centrality. ASE/IEEE International Conference on Social Computing (SocialCom)
- [12] Hamid, R., et al. 2009. A Novel Sequence Representation for Unsupervised Analysis of Human Activities, Artificial Intelligence.
- [13] Hastie, T., Tibshirani, R., and Friedman, J. 2008. *The Elements of Statistical Learning (2nd edition)*. Springer-Verlag.
- [14] Hempstalk, K., Frank, E., and Witten, I.H. 2008. One-class classification by combining density and class probability estimation. In: W. Daelemans et al. (Eds.), ECML PKDD 2008, Part I, LNAI 5211, pp. 505-519.
- [15] Lee, J-Y., et al. 2012. Fast Anomaly Detection Given Duplicates. CMU Computer Science Technical Reports.
- [16] Lindauer, B. and Glasser, J. 2013. Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data. *Workshop on Research for Insider Threat (WRIT)*, IEEE, San Francisco, May 24, 2013.
- [17] Mappus, R. and Briscoe, E. 2013. Layered Behavioral Trace Modeling for Threat Detection. IEEE Intelligence and Security Informatics.
- [18] Meinshausen, N. 2006. Quantile Regression Forests. *Journal of Machine Learning Research*, 7, 983-999.
- [19] Memory, A. et al. 2013. Context-Aware Insider Threat Detection. *Proceedings of the Workshop on Activity Context System Architectures*. Bellevue, WA.
- [20] Neal, R. and Hinton, G. E. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, (pp. 355-368), Kluwer Academic Publishers.
- [21] Raytheon Corporation 2010. SureView™ Proactive Endpoint Information Protection, DOI= [http://www.raytheon.com/capabilities/rtnwcm/groups/iis/documents/content/rtn\\_iis\\_sureview\\_datasheet.pdf](http://www.raytheon.com/capabilities/rtnwcm/groups/iis/documents/content/rtn_iis_sureview_datasheet.pdf)
- [22] Ridgeway, G. 2005. Generalized boosted models: A guide to the gbm package. DOI= <http://cran.r-project.org/web/packages/gbm/vignettes/gbm.pdf>.
- [23] Riedy, J. and Bader, D.A. 2013. Multithreaded Community Monitoring for Massive Streaming Graph Data. 7th Workshop on Multithreaded Architectures and Applications (MTAAP), Boston, MA, May 24, 2013.
- [24] Riedy, J., Bader, et al. 2011. Detecting Communities from Given Seeds in Social Networks. CSE Technical Report, GA Inst. of Technology.
- [25] Riedy, J., Meyerhenke, H., and Bader, D.A. 2012. Scalable Multi-threaded Community Detection in Social Networks. 6th Workshop on Multithreaded Architectures and Applications (MTAAP), Shanghai, China, May 25, 2012.
- [26] Senator, T. E. 2005. Multi-stage Classification. In ICDM '05 Proceedings of the Fifth IEEE International Conference on Data Mining Pages 386-393. IEEE Computer Society Washington, DC.
- [27] The White House 2012. Presidential Memorandum -- National Insider Threat Policy and Minimum Standards for Executive Branch Insider Threat Programs. Retrieved February 1, 2013 from <http://www.whitehouse.gov/the-press-office/2012/11/21/presidential-memorandum-national-insider-threat-policy-and-minimum-stand>
- [28] Young, W. T et al. 2013. Use of Domain Knowledge to Detect Insider Threats in Computer Activities. Workshop on Research for Insider Threat, *IEEE CS Security and Privacy Workshops*, San Francisco, May 24, 2013.
- [29] Zhou, Z-H. 2012. *Ensemble Methods: Foundations and Algorithms*, Chapman & Hall/CR