Mining for Insider Threats in Business Transactions and Processes

William Eberle and Lawrence Holder, *Members, IEEE*

***Abstract*—Protecting and securing sensitive information are critical challenges for businesses. Deliberate and intended actions such as malicious exploitation, theft or destruction of data, are not only harmful and difficult to detect, but frequently these threats are propagated by an insider. Unfortunately, current efforts to identify unauthorized access to information such as what is found in document control and management systems are limited in scope and capabilities. This paper presents an approach to detecting anomalies in business transactions and processes using a graph representation. In our graph-based anomaly detection (GBAD) approach, anomalous instances of structural patterns are discovered in data that represent entities, relationships and actions. A definition of graph-based anomalies and a brief description of the GBAD algorithms are presented, followed by empirical results using a discrete-event simulation of real-world business transactions and processes.**

I. Introduction

Every day there are reports of insider threats that affect an IT organization's network, systems and information. Recent reports have indicated that approximately 6% of revenues are lost due to fraud, and almost 60% of those fraud cases involve employees [10]. The Identity Theft Resource Center recently reported that 15.8 percent of security breaches so far in 2008 have come from insiders, up from 6 percent in 2007 [5]. Various insider activities such as violations of system security policy by an authorized user, deliberate and intended actions such as malicious exploitation, theft, or destruction of data, the compromise of networks, communications, or other IT resources, and the difficulty in differentiating suspected malicious behavior from normal behavior, have hampered business activities. IT organizations, responsible for the protection of their company's valuable resources, require the ability to mine and detect internal transactions for possible insider threats. Yet, most organizations spend considerable resources protecting their networks and information from the outside world, with little effort being applied to the threats from within.

For years, companies have been analyzing their business processes for the purposes of streamlining operations, discovering wasteful overhead, overcoming inefficiencies in production, etc. However, there have also been several efforts applied towards analyzing business processes for fraud detection, which has led to an increase in pertinent data mining activity. Most of these approaches have dealt with the visualization of business processes, such as VisImpact [12]. Some approaches have used data/audit logs that are collected by a company, in order to generate fraud alerts in near real-time. In addition, the ability to mine relational data has become important for detecting *structural* patterns. Recently there has been an impetus towards analyzing relational data using graph theoretic methods [2]. Graph-based data mining approaches analyze data that can be represented as a graph (i.e., vertices and edges). While there are approaches for using graph-based data mining for intrusion detection [3], little work has been done in the area of *graph-based anomaly detection*, especially for application to business processes, such as in document control and management systems.

In this paper, we present such an approach called Graph-Based Anomaly Detection (GBAD) [4]. GBAD discovers anomalous instances of structural patterns in data that represent entities, relationships and actions. Input to GBAD is a labeled graph in which entities are represented by labeled vertices and relationships or actions are represented by labeled edges between entities. Using the minimum description length (MDL) principle to identify the normative pattern that minimizes the number of bits needed to describe the input graph after being compressed by the pattern, GBAD uses algorithms for identifying the three possible changes to a graph: modifications, insertions and deletions. Each algorithm discovers those substructures that match the closest to the normative pattern without matching exactly. As a result, GBAD is looking for those activities that appear to match normal (or legitimate) transactions, but in fact are structurally different. This is a promising novel approach to discovering anomalies as most anomaly detection approaches use profiles based on non-structural attributes, and then search for outliers to those profiles [18].

Take for instance the document flow scenario of an order processing system, as shown in Figure 1. This example would consist of individual transactions where personnel

This material is based upon work supported by the Department of Homeland Security under Contract No. N66001-08-C-2030. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Homeland Security.

W. Eberle is with the Department of Computer Science, Tennessee Technological University, Cookeville, TN 38505 USA (e-mail: weberle@tntech.edu).

L. Holder is with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164 USA (e-mail: holder@wsu.edu).

receive, process and possibly pass on documents to other personnel or departments. However, when this information is represented as a graph, possible anomalous actions can be considered "additional structure" within the graph that was an unexpected deviation from the normal pattern of document flow. For example, the Sales department might also pass the Order Acknowledgement to another customer, providing "inside information."

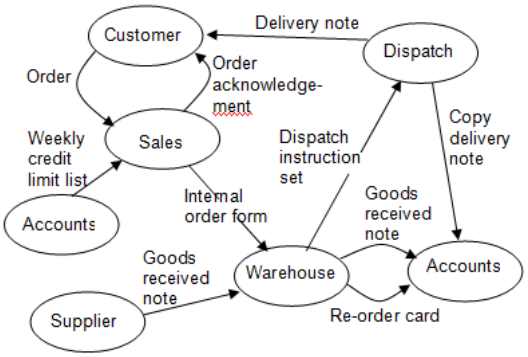

Figure 1. Order processing scenario.

This example will be discussed in more detail later in this paper.

In the following section, we present a definition of what is a *graph-based anomaly*, some assumptions that we have made, and a brief introduction to the three graph-based detection algorithms embodied in GBAD [6]. The primary contribution of this work is the applicability of these algorithms to the detection of insider threats in business transactions and processes. In this paper we present the flow of information associated with application processing, and evaluate GBAD's ability to discover deviations that could indicate attempted abuse. We then conclude with some discussion of future work.

II. GRAPH-BASED ANOMALY DETECTION

A. Definition

The idea behind the approach presented in this paper is to find anomalies in graph-based data where the anomalous substructure in a graph is part of (or attached to or missing from) a *normative substructure*.

Definition: *A graph substructure S' is anomalous if it is not isomorphic to the graph's normative substructure S, but is isomorphic to S within X%.*

X signifies the percentage of vertices and edges that would need to be changed in order for *S'* to be isomorphic to *S*. The importance of this definition lies in its relationship to any deceptive practices that are intended to illegally obtain or hide information. The United Nations Office on Drugs and Crime states the first fundamental law of money laundering as "The more successful money-laundering apparatus is in imitating the patterns and behavior of legitimate transactions, the less the likelihood of it being exposed" [1].

For a structural graph-based anomaly, there are several situations that might occur:

1. *A vertex exists that is unexpected.*
2. *An edge exists that is unexpected.*
3. *The label on vertex is different than was expected.*
4. *The label on edge is different than was expected.*
5. *An expected vertex is absent.*
6. *An expected edge between two vertices is absent.*

In essence, there are three general *categories of anomalies*: insertions, modifications and deletions. Insertions would constitute the first two situations; modifications would consist of the third and fourth situation; and deletions would categorize the last two situations.

B. Assumptions

Many of the graph-based anomaly detection approaches up to now have assumed that the data exhibits a power-law distribution [13]. The advantage of the approaches presented in this paper is that it does not assume the data consists of a power-law behavior. In fact, no standard distribution model is assumed to exist. All that is required is that the data is *regular*, which in general means that the data is "predictable". While there are many data sets that are not regular in nature, in general, business processes consist of transactions that would exhibit regular patterns of behavior. After all, that is why companies set up processes in the first place – to establish rules and guidelines for *normal* business activity.

In order to address our definition of an anomaly, we make the following assumptions about the data.

Assumption 1: *The majority of a graph consists of a normative pattern, and no more than X% of the normative pattern is altered in the case of an anomaly.*

Since our definition implies that an anomaly constitutes a minor change to the prevalent substructure, we would chose a small percentage (e.g., 10%) to represent the most a substructure would be changed in a fraudulent action.

Assumption 2: *Anomalies consist of one or more modifications, insertions or deletions.*

As was mentioned earlier, there are only three types of changes that can be made to a graph. Therefore, anomalies that consist of structural changes to a graph must consist of one of these types.

Assumption 3: *The normative pattern is connected.*

In the real-world scenarios of business transactions and processes, the entities are typically linked to each other in some way. Certainly, graphs could contain potential anomalies across disconnected substructures, but at this point, we are constraining our research to only connected anomalies.

C. Approach

Most anomaly detection methods use a supervised approach, which requires some sort of baseline of information from which comparisons or training can be performed. In general, if one has an idea what is normal behavior, deviations from that behavior could constitute an anomaly. However, the issue with those approaches is that one has to have the data in advance in order to train the system, and the data has to already be labeled (e.g., normal employee transaction versus threatening insider activity).

GBAD (Graph-based Anomaly Detection) [6] is an *unsupervised* approach, based upon the SUBDUE graph-based knowledge discovery method [7]. Using a greedy beam search and Minimum Description Length (MDL) heuristic [8], each of the three anomaly detection algorithms in GBAD uses SUBDUE to provide the top substructure, or normative pattern, in an input graph. In our implementation, the MDL approach is used to determine the best substructure(s) as the one that minimizes the following:

$$M(S,G) = DL(G \mid S) + DL(S)$$

where G is the entire graph, S is the substructure, $DL(G|S)$ is the description length of G after compressing it using S, and $DL(S)$ is the description length of the substructure.

We have developed three separate algorithms: GBAD-MDL, GBAD-P and GBAD-MPS. Each of these approaches is intended to discover all of the possible graph-based anomaly types as set forth earlier. The following is a brief summary of each of the algorithms, along with some simple business process examples to help explain their usage. The reader should refer to [6] for a more detailed description of the actual algorithms.

Information Theoretic Algorithm (GBAD-MDL)

The GBAD-MDL algorithm uses a Minimum Description Length (MDL) heuristic to discover the best substructure in a graph, and then subsequently examines all of the instances of that substructure that "look similar" to that pattern – or more precisely, are *modifications* to the normative pattern. In Noble and Cook's work on graph-based anomaly detection [9], they presented a similarly structured example (albeit with different labels) to the one shown in Figure 2.

In this example, the normal business process involves Sales sending an order to the Dispatcher, the Dispatcher verifying the order and sending in onto the Warehouse, and the Warehouse confirming the fulfillment of the order with Sales. When applying the GBAD-MDL algorithm to this example, the circled substructure in Figure 2 is reported as being anomalous. In this case, there are three entities communicating for each order, but Accounts is handling the order instead of Sales - going outside the normal process. With Noble and Cook's approach, the "Accounts" vertex would have correctly been shown to be the anomaly, but the importance of this new approach is that a larger picture is provided regarding its associated substructure. In other words, not only are we providing the anomaly, but we are also presenting the context of that anomaly within the graph (the individual anomaly within the instance is in **bold**.)

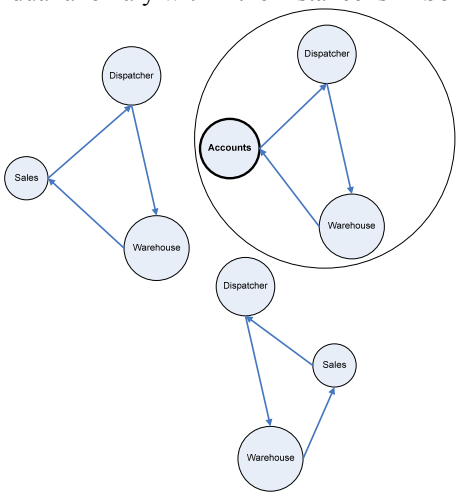

Figure 2. Example with anomalous instance circled.

Probabilistic Algorithm (GBAD-P)

The GBAD-P algorithm uses the MDL evaluation technique to discover the best substructure in a graph, but instead of examining all instances for similarity, this approach examines all extensions (or *insertions*) to the normative substructure with the lowest probability. The difference between the algorithms is that GBAD-MDL is looking at instances of substructures with the same characteristics (e.g., size), whereas GBAD-P is examining the probability of extensions to the normative pattern to determine if there is an instance that includes edges and vertices that are probabilistically less likely than other possible extensions. Taking the business process example again, Figure 3 shows the process flow between a warehouse (W), dispatcher (D), accounting (A) and the customer (C).

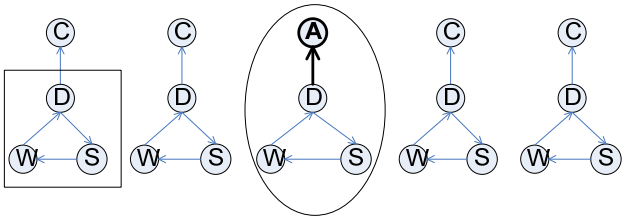

Figure 3. Example with instance of normative pattern boxed and anomaly circled

In this example, the normal process involves a communication chain between Sales, Warehouse and Dispatcher, with the order confirmation being conveyed by the Dispatcher to the Customer. After the first iteration of the GBAD-P algorithm, the boxed instance in Figure 3 is

one of the instances of the best substructure. Then, on the second iteration, extensions are evaluated, and the circled instance is the resulting anomalous substructure. In this example, the Dispatcher is communicating with Accounts when it should have been the Customer. Again, the edge and vertex (shown in **bold**) are labeled as the actual anomaly, but the entire anomalous substructure is output to provide additional context for possible analysis.

Maximum Partial Substructure (GBAD-MPS)

The GBAD-MPS algorithm again uses the MDL approach to discover the best substructure in a graph, then it examines all of the instances of parent (or ancestral) substructures that are missing various edges and vertices (i.e., *deletions*). The value associated with the parent instances represents the cost of transformation (i.e., how much change would have to take place for the instance to match the best substructure). Thus, the instance with the lowest cost transformation is considered the anomaly, as it is closest (maximum) to the best substructure without being included on the best substructure's instance list. If more than one instance have the same value, the frequency of the instance's structure will be used to break the tie if possible. Consider the slightly more complex graph of a business process, involving multiple transactions that are linked together by common entities, as shown in Figure 4.

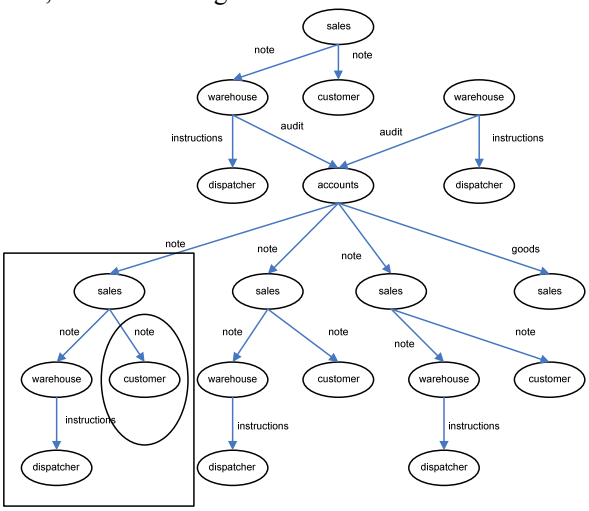

Figure 4. Example with instance of normative pattern boxed and anomaly circled.

In this example, the normative pattern in the process is a Sales person communicating with the Warehouse and a Customer, and the Warehouse corresponding with a Dispatcher. Suppose we take one of the instances of the normative pattern (shown in the box), and remove an edge and its associated vertex (shown in the circle). When applying GBAD-MPS to that modified graph, an anomalous substructure, similar to the normative pattern, is discovered, where the Customer entity is missing along with the "note" link from Sales.

Tests and Performance

In order to systematically test each of the algorithms, we created synthetic graphs of various sizes and connectivity. We then repeated experiments, each time randomly modifying, inserting or deleting (and sometimes a combination of all types) vertices and edges. We measured GBAD's ability to not only correctly identify the created anomalies (true positives versus false positives), but also its ability to not miss the anomaly. The overall results were that GBAD never found less than 95% of the anomalies, with minimal (none in most cases) false positives reported.

The average running times of the algorithms is anywhere from < 1 second to ~45 minutes, depending upon the size of the graphs. The larger the graph, as well as the number of subgraphs one wants to analyze for anomalous structure, the greater the runtime for the algorithms. In general, the running time of GBAD is polynomial in the size of the graph and the parameters of the algorithm. The ability to discover the anomalies is sometimes limited by the *resources* allocated to the algorithm. Given a graph where the anomalous substructure consists of the *minimal* deviation from the normative pattern, if a sufficient amount of processing time and memory is provided, all of these algorithms will discover the anomalous substructure with no false positives. However, the ability to discover anomalies (per our definition) is also hampered by the amount of *noise* present in the graph. The issue is that if noise is a smaller deviation from the normative pattern than the actual anomaly, it may score higher than the targeted anomaly (depending upon the frequency of the noise).

The reader should refer to [6] for more information regarding the experimental results and analysis of GBAD, including experiments on two diverse real-world data sets: cargo shipments and network traffic.

III. INTEGRATION OF GBAD AND OMNET++

In order to perform a systematic evaluation of the Graph-Based Anomaly Detection (GBAD) approach for identifying anomalies, or insider threats, in business transactions or processes, we used the OMNeT++ discrete event simulator [14] to model transactions and processes, generate transaction and process data, represent the data in graph form, and then analyze the graphs using GBAD. This process has two main benefits. First, we can model many different types of transactions with known structure and known anomalies, which allows us to easily verify GBAD's ability to detect these anomalies. Second, the OMNeT++ framework can be used to model real business processes to further evaluate the real-world applicability of the GBAD approach. Here we give a brief introduction of this process on a simple business transaction example, followed by a more complex example representing a known business process.

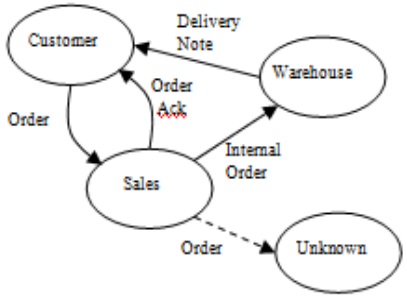

Figure 5. Depiction of an order fulfillment process; dashed arrow indicates a low-probability anomaly.

IV. BUSINESS TRANSACTIONS

Consider the order-fulfillment process depicted in Figure 5. The process is initiated by the Customer placing an Order, which is sent to the Sales department. The Sales department sends an Order Acknowledgement back to the Customer and sends an Internal Order to the Warehouse. Once the Warehouse ships the order, they send a Delivery Note to the Customer. One possible anomaly in this process is when someone in the Sales department copies the Order to an Unknown entity, perhaps to leak insider information to a competitor about the order.

```
simple Customer
  parameters:
    maxOrders: numeric const;
  gates:
    in: fromSales, fromWarehouse;
    out: toSales;
endsimple

simple Sales
  parameters:
    probAnomaly: numeric const;
  gates:
    in: fromCustomer;
    out: toCustomer, toWarehouse, toUnknown;
endsimple

simple Warehouse
  gates:
    in: fromSales;
    out: toCustomer;
endsimple

simple Unknown
  gates:
    in: fromSales;
endsimple

simple Utility
endsimple

module OrderProcess
  submodules:
    customer: Customer;
    sales: Sales;
    warehouse: Warehouse;
    unknown: Unknown;
    utility: Utility;
  connections:
    customer.toSales --> sales.fromCustomer;
    sales.toCustomer --> customer.fromSales;
```

Figure 6. The orderprocess.net file describes the different modules and how they interconnect.

First, we will see how this order-fulfillment process is defined within OMNeT++. Figure 6 shows the definition for this process using the NED (Network Description) language. Each node in the process is defined as a module in NED. Modules (like OrderProcess) can consist of sub-modules and their interconnections. The actual function of each module (how it processes messages) is defined in C++. The Utility module provides utility functions accessible by the other modules. After receiving an Order message, the Sales module waits 10-60 seconds and then sends an Order Acknowledgement message to the Customer module, sends an Internal Order message to the Warehouse module, and with a Bernoulli probability of 0.001 (as defined in the omnetpp.ini file) sends an Order message to the Unknown module.

The OMNeT++ user's manual describes the procedure for making and executing the simulation. Figure 7 shows a portion of the output from the order fulfillment simulation. In addition to the logging information produced by OMNeT++, the figure also shows the GBAD-related messages printed from each module describing order-related messages as they are sent and received by the modules. It is this information we will use to construct graphs of the ordering process. A utility program called "o2g" converts the GBAD-enhanced OMNeT++ simulation output into a Subdue-formatted graph.

For the experiment depicted in Figure 5, representing the processing flow of 1,000 orders, we generated a graph of approximately 3,000 vertices and 4,000 edges. From this graph, GBAD is able to successfully discover, with no false-positives, the anomaly shown with dotted lines and a larger font in Figure 8.

```
OMNeT++/OMNEST Discrete Event Simulation  (C)
1992-2005 Andras Varga
Release: 3.3, edition: Academic Public
License.
See the license for distribution terms and
warranty disclaimer
Setting up Cmdenv...

Preparing for Run #1...
Setting up network `orderprocess'...
Initializing...

Running simulation...
** Event #0 T=0.0000000 (0.00s).  (Customer)
orderprocess.customer (id=2)
[GBAD] 1 Order: Customer -> Sales (0)
** Event #1 T=0.0000000 (0.00s).  (Sales)
orderprocess.sales (id=3)
** Event #2 T= 30.8511 (30.85s).  (Sales)
orderprocess.sales (id=3)
[GBAD] 1 InternalOrder: Sales -> Warehouse
(30.8511)
[GBAD] 1 OrderAcknowledgement: Sales ->
Customer (30.8511)
** Event #3 T= 30.8511 (30.85s).  (Warehouse)
orderprocess.warehouse (id=4)
```

Figure 7. Partial OmNet++ simulation output.

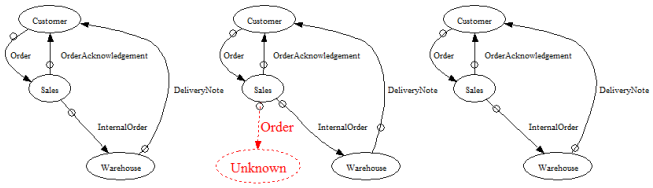

Figure 8. Subdue-formatted (partial) graph produced from GBAD-enhanced OmNet++ simulation output.

V. BUSINESS PROCESSES

Next, we simulated a document processing scenario that was motivated by two real-world sources of information. One source is the incidents reported in the CERT Insider Threat documents [15][16][17] that involve privacy violations in a government identification card processing organization and fraud in an insurance claim processing organization. Another source, for which our model directly simulates, is based on the process flow associated with a passport application [11]. The outline of this process flow, depicted in Figure 9, is as follows:

1. The applicant submits a request to the frontline staff of the organization.
2. The frontline staff creates a case in the organization's database and then submits the case to the approval officer.
3. The approval officer reviews the case in the database and then assigns the case to one of the case officers. By default, there are three case officers in the organization.
4. The assigned case officer reviews the case. The assigned case officer may request additional information from the applicant, which is submitted to the frontline staff and then forwarded to the assigned case officer. The assigned case officer updates the case in the database based on this new information. The assigned case officer may also discuss the case with one or more of the other case officers, who may review the case in the database in order to comment on the case. Ultimately, the assigned case officer will recommend to accept or reject the case. This recommendation is recorded in the database and sent to the approval officer.
5. Upon receiving the recommendation from the assigned case officer, the approval officer will make a final decision to accept or reject the case. This decision is recorded in the database and sent to both the frontline staff and the applicant.
6. Finally, upon receiving the final decision, the frontline staff archives the case in the database.

There are several scenarios where potential insider threat anomalies might occur, including:

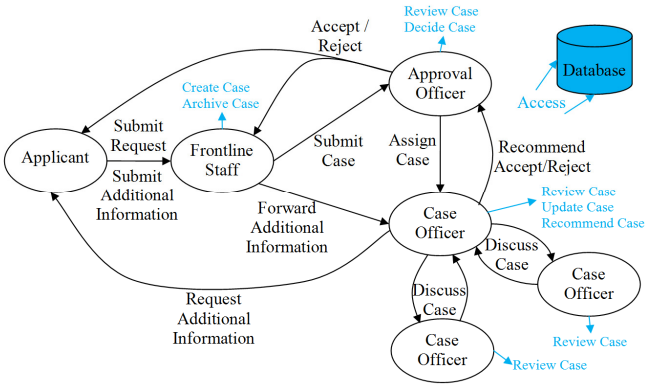

Figure 9. Information flow in request/claim approval scenario.

1. Frontline staff performing a Review Case on the database (e.g., invasion of privacy).
2. Frontline staff submits case directly to a case officer (bypassing the approval officer).
3. Frontline staff recommends or decides case.
4. Approval officer overrides accept/reject recommendation from assigned case officer.
5. Unassigned case officer updates or recommends case.
6. Applicant communicates with the approval officer or a case officer.
7. Unassigned case officer communicates with applicant.
8. Database access from an external source or after hours.

Representing the processing of 1,000 passport applications, we generated a graph of approximately 5,000 vertices and 13,000 edges, and proceeded to replicate some of the scenarios described above.

First, we randomly inserted an example that represents Scenario 1. While the GBAD-MDL and GBAD-MPS algorithms do not discover any anomalous structures, GBAD-P is able to successfully discover the one case out of 1,000 where a frontline staffer was performing a review of a case – a clear violation of their duties. Figure 10 shows the normative pattern and the anomalous edge "ReviewCase" between the "FrontlineStaff" node and the "Database" node.

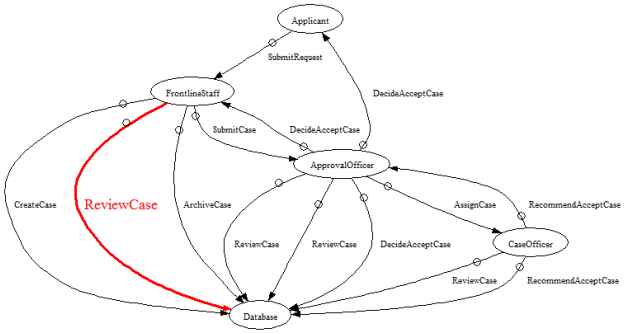

Figure 10. Scenario 1 normative pattern and anomaly.

The actual anomaly in Figure 10 is shown with a bolded edge and larger label font. Also, while not shown here, this same structural anomaly can be found in scenarios 3 and 6. Scenario 3 consists of an extra edge ("RecommendAcceptCase") going from the "FrontlineStaff" node to the "Database" node, and as such is only different from Scenario 1 by the label on the edge. Scenario 6 consists of an extra edge between the "Applicant" node and the "ApprovalOfficer" (or "CaseOfficer") node, which is structurally identical to the other two scenarios – an unexpected edge between two expected vertices.

For Scenario 2, we randomly inserted three examples where a frontline staffer submitted a case directly to a case officer, instead of sending it to the approval officer. In this case, GBAD-P and GBAD-MDL do not uncover any anomalous structures, whereas GBAD-MPS is able to successfully discover all three instances where the frontline staffer did not submit the case to the approval officer. Figure 11 shows the normative pattern and the missing "SubmitCase" edge between "FrontlineStaff" and "ApprovalOfficer", the missing "ReviewCase" edge between "ApprovalOfficer" and "Database", and the missing "AssignCase" edge between "ApprovalOfficer" and "CaseOfficer".

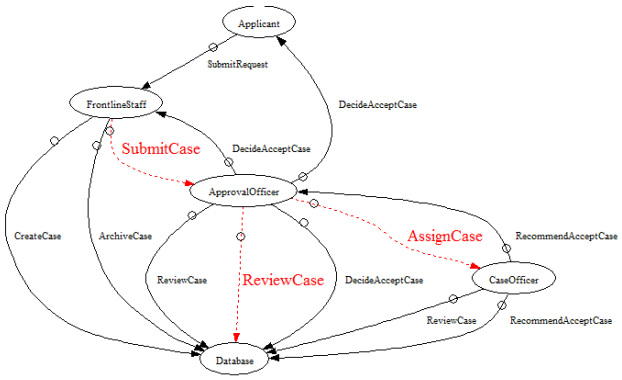

Figure 11. Graph of Scenario 2, showing the normative pattern and missing edges.

The actual anomalies in Figure 11 are shown with a larger label font and a dashed edge, indicating their absence from the graph.

For Scenario 4, we randomly modified three examples by changing the recommendation that the "CaseOfficer" sends to the "ApprovalOfficer". In one example, the "CaseOfficer" recommends to accept the case, and the recommendation from the "ApprovalOfficer" is changed to rejecting the case, and in the other two examples the reverse is implemented. For this example, GBAD-MDL and GBAD-MPS do not find any anomalies, and GBAD-P only discovers one of the anomalous examples (where the "CaseOfficer" recommends to reject the case but the "ApprovalOfficer" decides to accept the case. Figure 12 shows the normative pattern and the anomalous structures from this example.

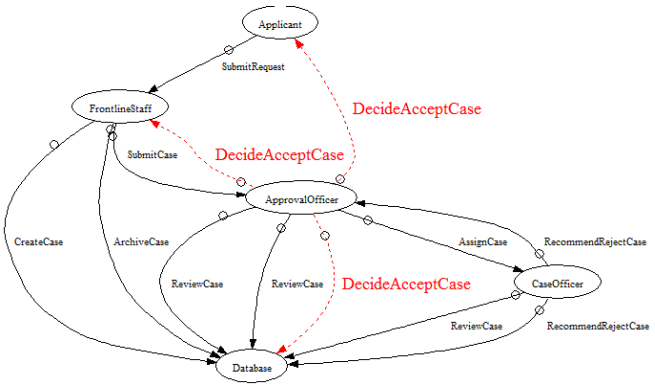

Figure 12. Graph of Scenario 4, showing the normative pattern and unexpected edge labels.

When we have GBAD report on the top two most anomalous substructures, instances of that type (reject changed to accept) are discovered, but we are still missing the first anomalous example (accept changed to reject). The issue is that we are dealing with multiple normative patterns (i.e., multiple substructures that can be considered normative in the entire graph.) In this case, there are two basic normative patterns – one where the "ApprovalOfficer" and "CaseOfficer" both accept a case, and one where the "ApprovalOfficer" and "CaseOfficer" both reject a case. However, when we modified the GBAD-P algorithm to analyze the top *N* normative patterns, both of the examples where the "CaseOfficer" recommends rejecting the case but the "ApprovalOfficer" accepts the case, are reported as the most anomalous examples, and the next most anomalous instance reported is the other anomalous example. Also, no other substructures were reported as anomalous along with these top three anomalies (i.e., no false positives).

For Scenario 5, we randomly inserted into two examples the situation where a "CaseOfficer" recommends to accept a case for which they were not assigned. In this scenario, GBAD-MDL does not report any anomalies, while both GBAD-MPS and GBAD-P each discover both anomalous instances. GBAD-MPS discovers the anomalies because the "CaseOfficer" has assigned himself to the case without any corresponding recommendation back to the "ApprovalOfficer" or "Database", while GBAD-P uncovers the extra "CaseOfficer" and his unauthorized assignment to the case. Figure 13 shows the normative pattern and the anomalous structures from one of these examples. Also, while not shown, this same structural anomaly can be found in scenario 7. Scenario 7 consists of an extra edge going from the unauthorized "CaseOfficer" node to the "Customer" node, and as such is only different from Scenario 5 by the label on the edge and the targeted node.

With the added aspect of time, found in Scenario 8, we are currently investigating the analysis of numerical attributes and how to incorporate them into the graph structure. Our initial analysis of discrete values is not included in this paper, and is being addressed in future work. It should also

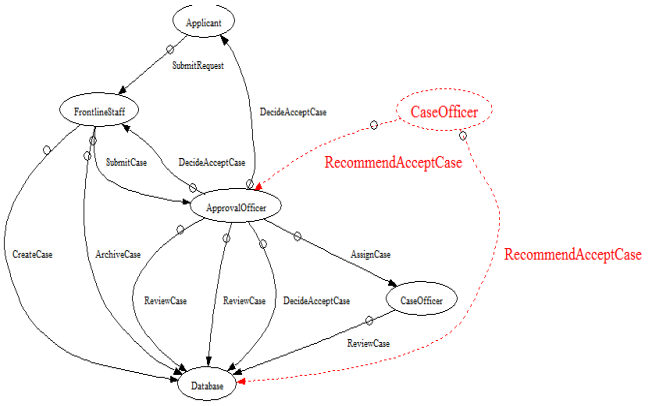

Figure 13. Scenario 5, unauthorized handling of a case.

be noted that the above process flows do not include the possible actions of a case officer requesting further information from the customer, or having a discussion with other case officers. That would involve further analysis of multiple normative patterns, and is something we will also be addressing in the future.

All of the above scenario graphs were rendered using the AT&T graph visualization program GraphViz (http://www.graphviz.org/).

VI. Conclusions and Future Work

The three algorithms presented in this paper are able to discover an anomaly when it consists of a small change to the normative pattern. Using the minimum description length principle and probabilistic approaches, we have been able to successfully discover anomalies in graphs and patterns of varying sizes with minimal to no false positives. Results from running the GBAD algorithms on simulated business transactions and processes have demonstrated the usefulness of applying these graph theoretic approaches. Some future directions that we are exploring include the incorporation of traditional data mining approaches as additional quantifiers to determining anomalousness, as well as applying graph theoretic algorithms to dynamic graphs that are changing over time. In addition, using the OMNeT++ example presented in this paper, we can create limitless numbers and varieties of simulations modeling business processes. These can then be used to evaluate GBAD both systematically and on models of real-world processes. The importance to this technology lies in the necessity of being able to detect insider threats before damage has been done. Organizations need advanced tools to detect insider threats, and they need to be able act in a timely fashion.

References

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

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

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

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

[5] L. Foley, "ITRC Beach Meter Reaches 342, to Date", *Reuters*, June 30, 2008.

[6] W. Eberle and L. Holder, "Anomaly Detection in Data Represented as Graphs," *Intelligent Data Analysis, An International Journal*, Volume 11(6), 2007.

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

[8] J. Rissanen, *Stochastic Complexity in Statistical Inquiry*, World Scientific Publishing Company, 1989.

[9] C. Noble and D. Cook, "Graph-Based Anomaly Detection," *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 631-636, 2003.

[10] *2006 AFCE Report to the Nation on Occupational Fraud & Abuse*, Association of Certified Fraud Examiners, 2006.

[11] A. Chun. "An AI framework for the automatic assessment of e-government forms," *AI Magazine*, Volume 29, Spring 2008.

[12] M. Hao, D. Keim, U. Dayal and J. Schneidewind. "Business process impact visualization and anomaly detection," *Information Visualization*, Volume 5, Issue 1, March 2006.

[13] M. Faloutsos, P. Faloutsos and C. Faloutsos. "On Power-law Relationships of the Internet Topology," *Proceedings of the conference on applications, technologies, architectures, and protocols for computer communications*, SIGCOMM, pp. 251-262, 1999.

[14] OMNeT++. *http://www.omnetpp.org/*.

[15] M. Randazzo, M. Keeney, E. Kowalski, D. Cappelli and A. Moore. "Insider Threat Study: Illicit Cyber Activity in the Banking and Finance Sector," *http://www.cert.org/insider_threat/*, 2004.

[16] E. Kowalski, D. Cappelli and A. Moore. "Insider Threat Study: Illicit Cyber Activity in the Information Technology and Telecommunications Sector," *http://www.cert.org/insider_threat/*, 2008.

[17] E. Kowalski, T. Conway, S. Keverline, M. Williams, D. Cappelli and A. Moore. "Insider Threat Study: Illicit Cyber Activity in the Government Sector," *http://www.cert.org/insider_threat/*, 2008

[18] V. Chandola, A. Banerjee and V. Kumar. "Anomaly Detection: A Survey," Technical Report, TR 07-017, University of Minnesota, August 15, 2007.