Stopping the Insider at the Gates: Protecting Organizational Assets Through Graph Mining*

Pablo Moriano[1]†, Jared Pendleton[2], Steven Rich[3], and L. Jean Camp[1]

[1] *School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN 47408, USA*
{pmoriano, ljcamp}@indiana.edu

[2] *Advanced Security Initiatives Group, Cisco Systems, Inc., Knoxville, TN 37932, USA*
jarpendl@cisco.com

[3] *Advanced Security Research Group, Cisco Systems, Inc., Knoxville, TN 37932, USA*
srich@cisco.com

Abstract

The increasing threat of insider attacks has resulted in a correlated increase in incentives to monitor trusted insiders. Measures of volumes of access, detailed background checks, and statistical characterizations of employee behaviors are all commonly used to mitigate the insider threat. These traditional approaches usually rely on supervised learning models or case studies to determine the critical features or attributes that can be used as indicators. Such approaches require labeled data for correct characterization of the threat. Yet regardless of the incentives to detect the insider threat, the incentives to share detailed labeled data on successful malicious insiders have proven inadequate. To address this challenging data environment, we developed an innovative approach that captures the temporal evolution of user-system interactions, to create an unsupervised learning framework to detect high-risk insider behaviors. Our method is based on the analysis of a bipartite graph of user and system interactions. The graph mining method detects increases in potential insider threat events following precipitating events, e.g., a limited restructuring. We apply our method to a dataset that comprises interactions between engineers and components in a software version control system spanning 22 years, and automatically detect statistically significant events. We find that there is statistically significant evidence for increasing anomalies in the committing behavior after precipitating events. Although these findings do not constitute detection of insider threat events per se, they reinforce the idea that insider operations can be motivated by the insiders' environment and detected with the proposed method. We compare our results with algorithms based on volume-dependent statistics showing that our proposed framework outperforms those measures. This graph mining method has potential for early detection of insider threat behavior from user-system interactions, which could enable quicker mitigation.

Keywords: Anomaly detection, insider threat, bipartite graph, graph mining, community structure, IBM Rational ClearCase

1 Introduction

Insiders are employees that must be trusted with access to sensitive information, and because of that trust can be a major threat. Insiders have compromised organizations in multiple domains including manufacturing [2], finance [3], government [4], and even scientific research [5]. Even worse, insiders

*This paper is an extended version of the work originally published at the 9th International Workshop on Managing Insider Security Threats (MIST'17) in conjuntion with the ACM Conference on Computer and Communications Security (CCS), Dallas, TX, USA, October 30, 2017 [1].

†Corresponding author: 901 E. 10th Street, Room 233, Bloomington, IN 47408, USA, Tel: +1 (812) 856-3960, Web: http://pmoriano.com

attacks are consistently catalogued as the most costly given the elevated privilege that insiders have in terms of trust and access [6]. This makes the insider issue one of the most challenging problems in computer security [7].

As with many other complex systems (e.g., the Internet, online social networks, and the brain), information systems consist of a large number of interacting elements (e.g., users, services, devices, files) in which the aggregate activity of the system cannot be derived by analyzing individual contributions, i.e., their aggregate behavior is nonlinear. Graphs, where nodes represent the elements and edges capture the interactions between the elements of the system, have been used across multiple domains to capture the interactions between the elements of complex systems [8, 9]. The use of graphs to study the structure of complex systems has revealed some plausible explanations for the emergence of collective behavior in these systems such as the understanding of regular and anomalous behavior [10]. In this work, we treat the malicious insider as an anomaly and use bipartite graphs to detect their anomalous behaviors.

The resulting focus on malicious patterns, as opposed to malicious nodes, implements an assumption that the malicious insider is not intrinsically hostile. Rather, malicious behaviors can emerge over time or in respect to specific conditions. Static graphical analysis is based on the analysis of graph snapshots and cannot integrate temporal patterns. In contrast, the study of temporal graphs, where information of single graph snapshots is aggregated, tends to reflect more accurately the evolution of the system as nodes and edges appear and disappear over time [11, 12]. The focus of this work is to understand the malicious behaviors over time rather than identifying the static malicious nodes.

To understand such complex systems, empirical data with detailed temporal information is a prerequisite. Correct temporal information is much more readily available as a source of ground truth than correctly labeled insider threat datasets. In the context of information systems, temporally annotated datasets are widely available thanks to the presence of user-system interaction logs. This enables the use of graph mining analytics for the understanding of anomalous behavior such as the one that insiders might pose [13, 14].

For the purposes of this paper, we characterize and detect anomalous events in an information system based on a centralized version control system[1]. We identify time intervals during which significant changes in the structure of the temporal graphs may correspond to functional change points, e.g., a precipitating event[2]. This problem has also been referred to as change point detection [16].

We model user-system interactions in a version control system as a temporal bipartite graph where interactions occur exclusively between two types of nodes, (i) users and (ii) software components[3]. Note that the edges in this graph are only between these two types of nodes [17]. A one-mode projection of this graph is the *user graph* in which two nodes (users) are connected if they have interacted at least once with the same component [18]. Our methodology includes studying the evolution of the one-mode user graph to identify topological properties that characterize the system's normal behavior. Among these observed properties, those that do not follow the norm of the regular pattern are assumed to indicate the presence of an anomalous event. Such an event may indicate a potential insider incident or, at least, an event that requires further investigation [19].

In particular, the user graph allows us to explore the impact of precipitating events in user-system interactions [20]. Precipitating events are key events that have the potential to trigger insiders to become

[1] A centralized version control system keeps the history of changes on a central server from which everyone requests the latest version of the work and pushes the latest changes to, e.g., Concurrent Versions System and IBM Rational ClearCase.

[2] A precipitating event corresponds to a large-scale event that causes concerning behaviors in employees and predisposed them to malicious actions. In this category, we include layoffs, significant restructuring, and plant or facility closure. The term was first used in the insider threat literature in Moore et al. [15].

[3] A software component is a software module that encapsulates a set of related functions or data, and it is part of a larger software system. For example, the TCP/IP software component of an operative system. Hereafter, we refer to software components as simply components.

a threat to their employer. We hypothesized that precipitating events impact the behavior of interactions between users and components in the version control system by changing patterns of committing behavior. To test this hypothesis, we model and compare the volume of interactions between users over similar or related components as opposed to non-related components over time. To capture sets of users with similar patterns of interaction, we rely on the notion of community structure to identify communities, or clusters, i.e., groups of nodes having higher probability of being connected to each other than to members of other groups [21]. We show that the volume of interactions between users that contribute to unrelated components increases when precipitating events are announced. This indicates the impact of precipitating events in increasing the likelihood of a change in the interacting behavior between users and components, which might be a signal to monitor before an insider attack is committed.

This is an extended version of a previous study that analyzed the impact of precipitating events in the committing behavior of employees in a software version control system [1]. In this article we expand our analysis to include density dependent statistics on the bipartite graphs and illustrate that they do not provide a useful detection signature that is correlated with the precipitating events. We offer clear links between the significance of graph properties and the activities of users. We also revisit the results and conclusions of our previous study.

2 Related Work

The analysis of the insider threat using temporal graph mining is informed by past research in the characterization of insider threats, anomaly detection in temporal graphs, and detection of insider threat using graph-based approaches. Here, we provide an overview of related works in these three areas.

2.1 Characterization of insider threats

Much of the research on insider threats have been on the characterization of insiders. In general, two different categorizations have been proposed to classify insiders. The first one focuses on the intention of the attack [22]. Under this categorization, insiders are classified as (i) malicious, where the insider intentionally causes a negative impact on the confidentiality, integrity, and availability of the information system; and (ii) non-malicious (accidental), where an insider, through action or inaction but no malicious intent, causes harm.

The second categorization is given with respect to the purpose of the attack [7]. With that definition in mind, two types of attacks are defined more precisely, including (i) a sabotage attack in which the insider is able to change the value of an artifact used in the computation of a process; and (ii) a data exfiltration attack in which the insider provides access to artifacts for entities that are not entitled to that access.

In addition to the previous two-tiered categorization, Nurse et al. proposed a unifying framework to characterize insiders based on the motivation behind malicious threats and the human factors related to the unintentional cases [20]. This framework is of particular importance not only because it leverages previous insider threat case studies, but also due to its analysis of behaviors that may lead to attacks and the types of attacks that may be executed. The factors that are proposed to this end encompass precipitating events and the motivation to attack.

2.2 Anomaly event detection in temporal graphs

There are five general approaches for the design of event detection algorithms in temporal graphs [12]. First, compression-based methods represent the graph in a different compact space using methods such as minimum description length (MDL) [23]. Anomalous events are detected when it is difficult to get a compressed representation of the graph. For example, Sun et al. proposed reducing the binary representation of the adjacency matrix of a graph so as to minimize the cost of encoding [24].

Second, decomposition methods analyze the spectral properties of a matrix representation of a graph stream by inspecting regular patterns associated to the eigenvalues and eigenvectors. An event is reported when there is low similarity between the principal eigenvector of the current graph and the aggregated graph during the previous time frame. The work by Akoglu and Faloutsos applied this idea on a mobile graph of users when inspecting a correlation matrix between pairs of nodes over a time interval [25].

Third, distance measure methods evaluate distance between graphs as a metric to identify anomalous events. The distance between consecutive graphs is computed based on changes in a specific structural property. Consecutive graphs with a significant distance between them should raise an alarm. The work by Koutra et al. explored this idea by comparing graph adjacency matrices of pairwise node affinities using a variation of the Euclidean distance [26].

Fourth, statistical methods are based on constructing statistical (parametric or non-parametric) models (e.g., graph likelihood or the distribution of the eigenvalues) to identify deviations from models. Anomalous events are identified by calculating the likelihood of appearance of a particular graph object, e.g., node, edge, subgraph, when a new graph is added to a graph sequence. For example, Aggarwal et al. proposed a method that quantifies the probability of rare edges appearing between subgraphs, allowing to pinpoint time intervals where this happens [27].

Finally, community-based methods focused on analyzing the formation of graph community structures. The idea behind this approach is to report an anomalous event whenever there is a significant change in any of the communities. The work by Duan et al. computed the similarity between the partition of nodes of incoming graphs and previous graph segments, i.e., a subset of a series of graphs. A similarity below a certain threshold indicates the occurrence of an anomalous event [28].

The method proposed in this work relies on the notion of graph community structure. For a comprehensive discussion about event detection methods in temporal graphs, we refer the reader to the survey led by Ranshous et al [12].

2.3 Insider threat detection using graph-based approaches

Graph mining techniques have also been used as a tool to understand and identify malicious actions by insiders. Eberle et al. proposed an approach to detect anomalous subgraphs with respect to the number of transformations that a subgraph will need in order to be a reference—the normative or best—subgraph [14]. The approach relies on MDL to quantify the number of required transformations as a criterion of decision [29]. The authors validated their approach using empirical data on a passport processing scenario. In particular, they were able to identify some bypassable steps in the process of getting a passport, which represents an anomalous structure of unseen edges.

To address the dynamic nature of empirical data, in a recent work, Eberle et al. introduced a method for pattern learning and anomaly detection in streams using parallel processing [30]. This work offers a considerable improvement on speedup compared to the previous approach by allowing the processing of dynamic data. The authors validate their approach on empirical data of embassy employee activity in which the threat was information leakage by employees.

Closer to our work, Kent et al. used the notion of bipartite graphs—by capturing interactions through authentication logs between users and computers—for assessing network authentication trust risk and cyber attack mitigation [31]. In particular, they examined the number of connected components (i.e., a subgraph in which any two nodes are connected to each other by a path) in the bipartite graph to assess potential risk of credential stealing and compromise within an enterprise network. They found that the increase in the number of connected components in the bipartite is associated with a reduction in the risk associated with credential theft and subsequent credential hopping within the network.

Of similar nature, Chen et al. proposed an unsupervised learning model based on social network analysis for detecting anomalous access in collaborative information systems [32]. Their approach relied on the quantification of pairwise similarities of nodes in a graph based on their interactions with particular subjects when interactions are made between users and subjects in a bipartite graph setting. The authors validated their results with patient record access data and Wikipedia edit logs.

Note that the previous methods of insider threat detection (using graph mining techniques) were based on identifying anomalous graph structures (i.e., nodes, edges, subgraphs). The focus of our paper is on the detection of anomalous events (i.e., time intervals with an unusual pattern of interactions) on temporal bipartite graphs.

3 Methods

In this section, we detail the mathematical frameworks and data sources that were used to perform the analysis. We start by describing the temporal framework used to build the graphs (Section 3.1); the bipartite graph modeling (Section 3.2); the one-mode projection abstraction (Section 3.3); the detection problem definition (Section 3.4); the algorithm performance abstraction (Section 3.5); the metric of algorithm performance (Section 3.6); the proposed algorithm (Section 3.7); and the dataset used to arrive at the results (Section 3.8).

Our method builds graphs of user-system interactions and uses these to identify anomalous patterns. Anomalies are identified when engineers interact with multiple components where there is no history of interaction, particularly where none of their team members are interacting or have a history of interaction with those components. Performance is measured by the ability of the algorithm to detect increases in anomalous behavior after precipitating events.

3.1 Temporal abstraction

Consider the sequence of n intervals $A = \{A_1, A_2, \dots, A_n\} = \{A_k\}_{k=1}^{n}$, where

1. $A_k = [a_k, a'_k)$ for all $k < n$ and $A_n = [a_n, a'_n]$ for $k = n$;

2. $a_k < a'_k = a_{k+1}$ for all k; and

3. $a'_k - a_k = a'_\ell - a_\ell$ for all k, ℓ

An interval represents a fixed-length unit of time, e.g., a day of data. Condition (1) implies that all intervals are left-closed and right-open (except the last one which includes a'_n). It guarantees that the sequence of intervals is disjoint. Condition (2) implies that intervals are non-empty. Note that a'_k and a_{k+1} represent the time instants of a transition between intervals. For any interval A_k, the right endpoint a'_k corresponds to the left endpoint of the interval A_{k+1}. Together with Condition (1), Condition (2) guarantees that the union of all intervals $\bigcup_{k=1}^{n} A_k = [a_1, a'_n]$ is a closed interval. Finally, Condition (3) requires that any two intervals are of equal length.

3.2 Bipartite graph abstraction

A bipartite graph is a graph with two types of nodes. One type of node represents the original nodes (top nodes), while the other represents the groups with which they interact (bottom nodes) [33].

Let $\mathscr{H}_\top$ be the set of top nodes (e.g., the set of engineers). Similarly, let $\mathscr{H}_\bot$ be the set of bottom nodes (e.g., the set of components). Note that $\mathscr{H}_\top$ and $\mathscr{H}_\bot$ are disjoint sets of nodes. Furthermore, let $\mathscr{V}(k) \subseteq \mathscr{H}_\top \cup \mathscr{H}_\bot$ be the subset of nodes that interact (i.e., engineers and components) during interval $A_k = [a_k, a'_k)$. Let $\mathscr{W}(k) = \{\Omega_{ij}(k) : (i,j) \subseteq \mathscr{H}_\top \times \mathscr{H}_\bot\}$ be the incidence matrix of weights

$\Omega_{ij}(k)$ that captures the number of interactions between node i and node j during interval A_k. Let $\mathscr{G}(k) = (\mathscr{V}(k), \mathscr{W}(k))$ represent a weighted bipartite graph that captures all interactions that occur from endpoints a_k to a'_k, $k \in \{1, 2, \ldots, n\}$. Note that we do not differentiate between dynamics within an interval. The sequence $\{\mathscr{G}(k)\}_{k=1}^{n}$ denotes the bipartite graph series G.

3.3 One-mode projection abstraction

Bipartite graphs can be projected to one-mode projection graphs (with nodes of just one type). Let $\mathscr{G}_\top(k) = (\mathscr{H}_\top(k), \mathscr{W}_\top(k))$ be the top projection of $\mathscr{G}(k)$. Two nodes of $\mathscr{H}_\top(k)$ are connected if they have at least one neighbor in common in $\mathscr{G}(k)$, i.e., $\mathscr{W}_\top(k) = \{\omega_{uv}(k) : u, v \subseteq \mathscr{H}_\top\}$, where

$$\omega_{uv}(k) = \textstyle\sum_{r=1}^{|\mathscr{H}_\perp|} \Omega_{ur}(k) + \Omega_{vr}(k)$$

The sequence $\{\mathscr{G}_\top(k)\}_{k=1}^{n}$ denotes the top one-mode projection graph series $G_\top$. Correspondingly, the bottom projection $\mathscr{G}_\perp(k) = (\mathscr{H}_\perp(k), \mathscr{W}_\perp(k))$ is defined dually as it is illustrated in Figure 1. The sequence $\{\mathscr{G}_\perp(k)\}_{k=1}^{n}$ denotes the bottom one-mode projection graph series $G_\perp$. In the rest of this paper, we devote our study in terms of $G_\top$ which is the one-mode projection graph of user-system interactions, i.e., the projection in which nodes are exclusively engineers.

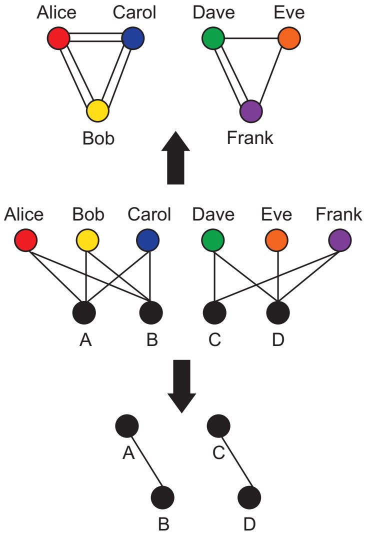

Figure 1: Bipartite graph abstraction. The top panel represents the engineer projection. The middle panel represents the original bipartite graph. The bottom panel represents the component projection.

3.4 Detection problem

We use $G_\top$, which captures the dynamics across intervals A_k, $k \in \{1, 2, \ldots, n\}$, as the basis for defining the anomaly event detection problem. In doing so, we evaluate the outcomes of anomaly detection by measuring structural properties with respect to the cumulative one-mode graph segment of length $m \in \mathbb{Z}^+$ defined as

$$\begin{aligned} \mathscr{G}_\top^m(k) &= (\mathscr{V}_\top^m(k), \mathscr{W}_\top^m(k)) \\ &= \bigoplus_{k'=k-m+1}^{k} \mathscr{G}_\top(k') = \mathscr{G}_\top(k-m+1) \oplus \cdots \oplus \mathscr{G}_\top(k) \end{aligned}$$

where

$$\mathscr{V}_{\top}^{m}(k) = \bigcup_{k'=k-m+1}^{k} \mathscr{V}_{\top}(k') \text{ and } \mathscr{W}_{\top}^{m}(k) = \sum_{k'=k-m+1}^{k} \mathscr{W}_{\top}(k')$$

For example, if $m = 7$, we aggregate data to form weekly graph segments.

Let lm, where $l \in \mathbb{Z}^{+}$ represents the smallest interval at which we evaluate the outcomes of anomalous detection (called the detection resolution). Note that if $l > 1$, then the intervals at which the graph segments are evaluated are not the same as the ones at which they are formed. The finest detection granularity satisfies $l = 1$, i.e., when the detection resolution is the same as the graph segment formation intervals. A larger value of l reflects that anomalous events are captured by the aggregation of consecutive graph segments. For instance, if $l = 2$, then an algorithm for detection aims to determine whether such an event occurs within intervals $(a_{k-lm+1}, a'_{k}] = (a_{k-2m+1}, a'_{k}]$, $k \in \{2m, \ldots, n\}$. Let $\bar{n} = \lfloor \frac{n}{lm} \rfloor$ be the total number of times the algorithm with resolution lm has to decide whether an event occurs. Let the set $E \subseteq \{1, 2, \ldots, \bar{n}\}$ represent the intervals at which at least one event occurs. The detection problem is specified as follows.

Given:

- (i) A one-mode projection graph series $G_{\top} = \{\mathscr{G}_{\top}(k)\}_{k=1}^{n}$; and
- (ii) A detection resolution $1 \leq lm < n$.

We want to:

- (ii) Design a detection algorithm that identifies the subset of intervals $\hat{E} \subseteq E$ in which at least one anomalous event occurs.

Condition (i) requires that the dataset can be modeled as a series of one-mode projection graphs that aggregate the interactions occurring during each interval. Condition (ii) assumes that a parameter can be selected to enable detection of anomalies at a desired timescale.

3.5 Algorithm performance abstraction

Consider a sequence of detection intervals $B = \{B_1, B_2, \ldots, B_{\bar{n}}\} = \{(a_{(t-1)lm+1}, a'_{tlm}]\}_{t=1}^{\bar{n}} = \{B_t\}_{t=1}^{\bar{n}}$. To measure performance, the output of the detection algorithm $\hat{E}$ is mapped into the sequence of intervals B. Let $\hat{e} \in \hat{E}$ be the index of a detection interval that is denoted as anomalous by the detection algorithm (i.e., the algorithm indicates the occurrence of at least one anomalous event within the interval). The set $\hat{E}$ can be represented by the indicator vector

$$\hat{O} = \bigvee \{\mathbb{1}_{B_t}(lm\hat{e}),\ \forall t \in \{1, 2, \ldots, \bar{n}\}\},\ \forall \hat{e} \in \hat{E}$$

where $\bigvee$ represents the OR operator and $\mathbb{1}_{B_t}(lm\hat{e})$ denotes the indicator function

$$\mathbb{1}_{B_t}(lm\hat{e}) = \begin{cases} 1 & \text{if } lm\hat{e} \in B_t \\ 0 & \text{if } lm\hat{e} \notin B_t \end{cases}$$

In other words, if $\mathbb{1}_{B_t}(lm\hat{e}) = 1$, the algorithm identifies an anomalous event in the detection interval $(a_{(t-1)lm+1}, a'_{tlm}]$ and labels it as an anomalous interval. The indicator vector $\hat{O}$ describes the interval indices, i.e., $t \in \{1, 2, \ldots, \bar{n}\}$ that contain an anomalous event.

Moreover, to characterize the occurrence of actual events during an interval, we define $e \in E$ as the index of a detection interval that is anomalous based on the ground truth. Let the indicator vector $O = \bigvee \{\mathbb{1}_{B_t}(lme),\ \forall t \in \{1, 2, \ldots, \bar{n}\}\},\ \forall e \in E$ represents the intervals that are anomalous based on the ground truth, i.e., the distribution of the anomalous events over the set of the $\bar{n}$ detection intervals. Figure 2 illustrates the proposed modeling framework. For example, suppose that $E = \{s, \bar{n}\}$ (represented by the horizontal arrows) and $\hat{E} = \{s\}$ (represented by the horizontal crossed arrow). To pinpoint the detection interval s, there might exist a time index $r = ms$ such that $\mathbb{1}_{B_s}(r) = 1$. This is represented by the vertical arrows in Figure 2.

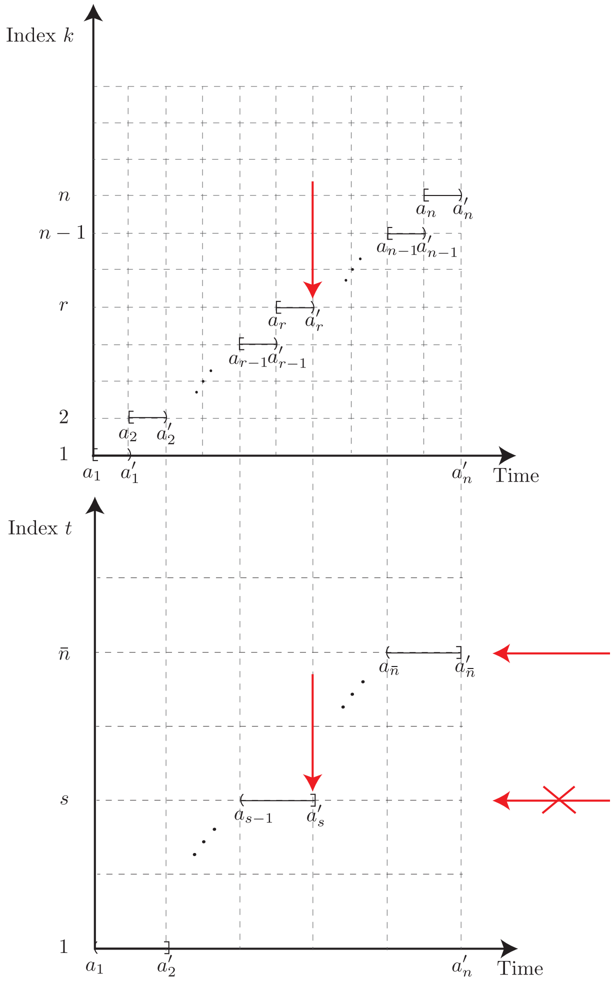

Figure 2: Abstraction of the detection problem. The top panel refers to the sequence of intervals that are used to build the graphs (here the graph formation interval $m = 1$). The bottom panel illustrates the aggregation of intervals to evaluate the performance of the detection algorithm (here detection resolution $lm = 2$). The vertical arrows represent the location of an anomalous event in both temporal representations. The horizontal arrows illustrate the sets E and $\hat{E}$.

3.6 Algorithm performance measure

The performance of a detection algorithm is measured based on identifying the anomalous detection intervals. Specifically, the performance of an algorithm is specified based on the set of time intervals $\hat{E}$ reported as anomalous by the detection algorithm and the set of time intervals E in which anomalies occur (ground truth).

We compare the performance of the detection algorithms using the true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) of the detection results. In particular, TP $= O \cdot \hat{O}$, FP $= O' \cdot \hat{O}$, FN $= O \cdot \hat{O}'$, and TN $= O' \cdot \hat{O}'$ where the symbol "$\cdot$" represents the dot product between two vectors, and O' and $\hat{O}'$ represents the complement of O and $\hat{O}$ respectively.

In other words, a detection algorithm specifies the intervals based on a detection criterion. Similarly, to measure performance, it is necessary to know the ground truth anomalous events. The detailed pseudo-code for the algorithm's performance measure is presented in Algorithm 1. Next, we introduce a detection criterion based on the dynamics of the formation of communities and the interaction of engineers across and within them.

Algorithm 1 Algorithm-Performance ($\hat{E}$, E, $\bar{n}$)

```
$\hat{O} \leftarrow zeros(\bar{n})$
for $\hat{e} \in \hat{E}$ do
    $\hat{O}_{\hat{e}} \leftarrow \{\}$
    for $t \in \{1,2,\ldots,\bar{n}\}$ do
        $\hat{O}_{\hat{e}} \leftarrow \hat{O}_{\hat{e}} \cup \mathbb{1}_{B_t}(\hat{e})$
    end for
    $\hat{O} \leftarrow \hat{O}$ OR $\hat{O}_{\hat{e}}$ (element wise)
end for
$O \leftarrow zeros(\bar{n})$
for $e \in E$ do
    $O_e \leftarrow \{\}$
    for $t \in \{1,2,\ldots,\bar{n}\}$ do
        $O_e \leftarrow O_e \cup \mathbb{1}_{B_t}(e)$
    end for
    $O \leftarrow O$ OR $O_e$ (element wise)
end for
$O' \leftarrow$ NOT $(O)$
$\hat{O}' \leftarrow$ NOT $(\hat{O})$
TP $\leftarrow O \cdot \hat{O}$
FP $\leftarrow O' \cdot \hat{O}$
FN $\leftarrow O \cdot \hat{O}'$
TN $\leftarrow O' \cdot \hat{O}'$
return (TP, FP, FN, TN)
```

3.7 Proposed algorithm

The proposed algorithm aims to define detection signatures based on deviations from the regular process of community interaction. To do so, we explore whether variations in the number of edges across communities (with respect to the total number) are indicators of anomalous events. This is done by comparing edges in the user graph with respect to a community partition reference over aggregate data.

Let the initial cumulative one-mode graph segment of length m_0, $1 \ll m_0 \ll n$ be defined as

$$\begin{aligned} \mathscr{G}_\top^{m_0} &= (\mathscr{V}_\top^{m_0}, \mathscr{W}_\top^{m_0}) \\ &= \bigoplus_{k'=1}^{m_0} \mathscr{G}_\top(k') = \mathscr{G}_\top(1) \oplus \cdots \oplus \mathscr{G}_\top(m_0) \end{aligned}$$

where $\mathscr{V}_\top^{m_0} = \bigcup_{k'=1}^{m_0} \mathscr{V}_\top(k')$ and $\mathscr{W}_\top^{m_0} = \sum_{k'=1}^{m_0} \mathscr{W}_\top(k')$.

The proposed detection algorithm requires the following assumption:

(A1) The initial cumulative graph segment $\mathscr{G}_\top^{m_0}$ can be naturally divided in non-overlapping communities, i.e., groups of nodes that can be grouped into subsets such that each set of nodes is densely connected internally and in which nodes belong to a single group [34].

Let the set $T = \{m_0 + m, m_0 + 2m, \ldots, \bar{n}m\}$ captures the time intervals at which the algorithm will be applied. Note that for $k \in T$, the series $\{\mathscr{G}_\top^m(k)\}$ forms a set of non-overlapping cumulative graph segments. The proposed algorithm pinpoints anomalous events by measuring the proportions of inter- and intra-community edges of the graph $\mathscr{G}_{\top}^{m}(k)$ with respect to the community partition of $\mathscr{G}_{\top}^{m_0}$, i.e., we want to identify the set $\hat{E}$ based on the diversification of community edges. Figure 3 shows a characterization of that situation.

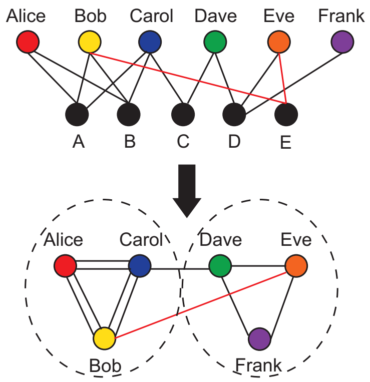

Figure 3: Malicious activity in the bipartite graph. The top panel represents an unusual interaction between Bob and Eve with component "E." The bottom panel represents the corresponding one mode projection graph with the anomalous edge crossing communities.

To do so, let $C(\mathscr{G}_{\top}^{m_0}) = \{0, 1, \ldots, c\}$ be a set of unique community identifiers where $c+1$ is the total number of detected communities in the initial cumulative graph segment $\mathscr{G}_{\top}^{m_0}$. The community to which engineer $i \in \mathscr{V}_{\top}^{m}(k) \cap \mathscr{V}_{\top}^{m_0}$ is assigned (with respect to $\mathscr{G}_{\top}^{m_0}$) is given by $c_i(k) : i \rightarrow C(\mathscr{G}_{\top}^{m_0})$. We computed the community partition of the initial cumulative graph segment using the Infomap algorithm [35]. Following similar ideas as in [36], let the set of inter-community edges be $I_{\curvearrowright}(\mathscr{G}_{\top}^{m}(k)) = \{(u,v) : \omega_{uv}(k) > 0 \wedge (c_u(k) \cap c_v(k)) = \varnothing\}$ and intra-community edges be $I_{\circlearrowright}(\mathscr{G}_{\top}^{m}(k)) = \{(u,v) : \omega_{uv}(k) > 0 \wedge (c_u(k) \cap c_v(k)) \neq \varnothing\}$. We also define the inter- and intra-community ratio as

$$c_{\curvearrowright}^{m}(k) = \frac{|I_{\curvearrowright}(\mathscr{G}_{\top}^{m}(k)|}{|I_{\curvearrowright}(\mathscr{G}_{\top}^{m}(k))| + |I_{\circlearrowright}(\mathscr{G}_{\top}^{m}(k))|} \tag{1}$$

$$c_{\circlearrowright}^{m}(k) = \frac{|I_{\circlearrowright}(\mathscr{G}_{\top}^{m}(k)|}{|I_{\curvearrowright}(\mathscr{G}_{\top}^{m}(k))| + |I_{\circlearrowright}(\mathscr{G}_{\top}^{m}(k))|} \tag{2}$$

respectively.

In particular, we are interested in identifying time intervals k, where $c_{\circlearrowright}^{m}(k) - c_{\curvearrowright}^{m}(k)$ is below median-3σ or above median+3σ. The median is used instead of the mean because this measure (over the entire period of study) cannot be assumed follow a normal distribution since appropriate hypothesis testing demonstrates that the normal distribution is not a good candidate to model the generation of the empirical observations. We used the interquartile range to estimate σ as it has been studied by others, e.g., [37]. The detailed pseudo-code for this algorithm is shown in Algorithm 2.

For algorithm performance comparison purposes, we replace the computation of $c_{\circlearrowright}^{m}(k) - c_{\curvearrowright}^{m}(k)$ by the respective graph topological property, e.g., nodes, edges, connected components, average degree, maximum degree, or maximum weight with respect to $\mathscr{G}_{\top}^{m}(k)$.

3.8 Dataset

IBM Rational ClearCase (hereafter ClearCase) is an enterprise-grade software configuration management system. Among its main features, it provides version control functionalities to large- and medium-size organizations allowing them to track software projects with thousands of developers. As of the date

Algorithm 2 Event-Detection ($G_\top$, m_0, m)

```
Compute community partition of $\mathscr{G}_\top^{m_0}$
$Y \leftarrow \{\}$                                        ▷ Array of intra−inter ratio samples
for $k$ in $\{m_0+m, m_0+2m, \ldots, \bar{n}m\}$ do
    Build $\mathscr{G}_\top^m(k) = \bigoplus_{k'=k-m+1}^{k} \mathscr{G}_\top(k')$
    Compute $I_\curvearrowright(\mathscr{G}_\top^m(k))$
    Compute $I_\circlearrowright(\mathscr{G}_\top^m(k))$
    Calculate $c_\curvearrowright^m(k)$ and $c_\circlearrowright^m(k)$ using eqs. 1 and 2
    $Y \leftarrow Y \cup \{c_\circlearrowright^m(k) - c_\curvearrowright^m(k)\}$
end for
median $\leftarrow \hat{F}_Y^{-1}(0.50)$                   ▷ $\hat{F}$ means the empirical CDF
$\delta \leftarrow \hat{F}_Y^{-1}(0.75) - \hat{F}_Y^{-1}(0.25)$          ▷ The interquartile range
$\hat{E} \leftarrow \{\}$
for $k$ in $\{m_0+m, m_0+2m, \ldots, \bar{n}m\}$ do
    if $Y(k) <= (\text{median} - 3\sigma)$ or $Y(k) >= (\text{median} + 3\sigma)$ then
        $\hat{E} \leftarrow \hat{E} \cup \{k\}$
    end if
end for
return $\hat{E}$
```

of this writing, ClearCase has a market share of about 2.5% among software configuration management competitors with 55% of their customers in the U.S. [38].

The ClearCase dataset analyzed in this paper comprises the complete activity between engineers and components in a major computer software enterprise. Components are software packages that encapsulate a set of related functions and store metadata allowing version control. In particular, we used data that spans 22 years from May 4, 1992 to March 23, 2014. We extracted the data from the source code base management database. Instances with no reference to the engineer or component name were not taken into account in this analysis. These comprised a negligible percentage of instances, i.e., on the order of $8{\times}10^{-6}$. (Thus the number of interactions in this dataset that were not captured in our graph mining method is sufficiently small that manual examination for insider activity would be quite feasible.)

Using this dataset, we built bipartite graphs to capture the interactions between engineers and components. In this bipartite graph, nodes are represented exclusively by engineers and components. Edges in the bipartite graph represent interactions, i.e., any type of activity that engineers have with components, including: commit a file, create a file, delete a file, create a branch, tag a branch, sync a branch, and collapse a branch. We did not differentiate between these different interactions and treat them as the same type of edges.

The dataset comprises $10,253$ distinct engineers, $1,729$ distinct components, and $12,577,667$ interactions during the observation period. Remember that our hypothesis is grounded on the idea that precipitating events might lead to structural changes in the committing behavior of engineers. With that in mind, Table 1 summarizes the details of the incidents used in this study, i.e., precipitating events that were announced and validated internally by the enterprise. These events correspond to limited restructuring events and have and effect in all business units of the enterprise.

Table 1: Summary of precipitating events during the observation period.

Event ID	Date	Jobs impacted	% affected employees
①	2001-04-16	8500	22.4
②	2011-07-18	6500	9.1
③	2012-07-23	1300	1.9
④	2013-03-26	500	0.7
⑤	2013-08-09	4000	5.3

4 Results

In this section, we present the results of the analyses on the bipartite graphs and the one-mode projection (or user graph) of user-system interactions. In the following analysis, our unit of time reference is the day, i.e., the scale of the variable k. To estimate the length of the window m (the window length that we use to accumulate interactions among engineers), we relied on the methodology proposed by [39], which estimated that the size of an observable window for a rigorous characterization of graph properties is at least one week, i.e., $m = 7$ days. This means that we build the bipartite and one-mode projection graphs by aggregating data over non-overlapping windows of 7 days (every week starting on Monday).

We compare the results of the proposed event detector framework to random chance. The purpose of this comparison is to ensure that the phenomena we identify are not a result of noise or simply the result of having stochastic data. We then compare our approach with metrics that are based on the volume of interactions. That is, we test if the proposed approach identifies insider risk more accurately than those that identify employees by frequency or intensity of access. Sheer counts of access are a core component of risk-based or accounting-based insider threat approaches. The model proposed here is more accurate and more precise. The model also offers fewer false negatives (i.e., higher recall).

We used the same visualization conventions for every plot. The blue solid lines show the raw data. Recall that the raw data corresponds to the empirical measures for each graph topological property. Dashed black lines represent the dates of the precipitating events listed in Table 1 with their corresponding label in a circle.

The results of these show that the precipitating events cannot be distinguished from other events using simple graph-based statistics. Our assumption is that although individual events, such as economic stress, may result in an individual becoming an insider threat, only systematic organizational changes should be correlated with large-scale increases in insider threat behaviors. Section 4.1 shows the graph-based statistics from the bipartite graphs. Section 4.2 shows the graph-based statistics for the one-mode projection graphs. Section 4.3 illustrates the algorithm evaluation using graph-based measurements and the proposed metric in this paper. In contrast with the results of graph-based measurements, we provide statistically significant evidence of detection of suspicious interactions after precipitating events have been announced using the proposed metric. Section 4.4 describes the way in which we obtained the results of the randomly generated algorithm and the performance comparison of each metric based on different detection resolutions.

In the following two sections we report basic graph properties. The purpose of this is to illustrate that naive application of graph mining on bipartite graphs without inclusion of community dynamics is an inadequate indicator of insider threats.

4.1 Bipartite graph properties series

We report measures related to the number of nodes, edges, connected components, average degree, maximum degree, and maximum weight for the bipartite graphs. Formalisms about the framework to build the bipartite graphs are defined in Section 3.2. The specific properties that we measured from these graphs are listed here for the reader. Note that edges only capture interactions between engineers and components. The number of nodes is $|\mathscr{V}(k)|$, which is the total number of engineers and components. The degree of node i is $d_i(k)$, i.e., its number of neighbors. The degree of a node is either the number of components that are touched by a single engineer or the number of engineers that touch a component. The set of edges of the graph $\mathscr{G}(k)$ is $\mathscr{E}(k)$ is the total number of unique component/engineer interactions. The number of edges is $|\mathscr{E}(k)| = \sum_{i \in \mathscr{V}(k)} d_i(k)/2$, which means this represents total number of interactions in the bipartite graph, i.e., system activity. A connected component is a subgraph in which any two nodes are connected to each other by paths. This means subgraphs can be the result of connections or similarities in connections between engineers and/or components so that any two vertices are connected by any path of interactions. The average degree of graph $\mathscr{G}(k)$ is $2 \times |\mathscr{E}(k)|/|\mathscr{V}(k)|$. This means the average number of interactions that nodes have reflects the fact that there are two entities in an interaction. The maximum degree of graph $\mathscr{G}(k)$ is the maximum number of neighbors in the graph, i.e., $\max\{d_i(k), \forall i \in \mathscr{V}(k)\}$. This is the maximum number of interactions among the nodes (either with engineers or components). The maximum weight of a graph $\mathscr{G}(k)$ is the maximum weight among the edges in the graph, i.e., $\max\{\omega_{ij}(k), \forall i, j \in \mathscr{V}(k)\}$. This means the maximum weight value among the interactions, i.e., the interaction with the higher intensity.

Figure 4 (top) shows the observed number of nodes (i.e., engineers and components) in the bipartite graphs. Figure 4 (middle) shows the number of unique edges representing the number of interactions between engineers and/or components. Figure 4 (bottom) shows the number of connected components in the bipartite graphs. There is a constant increase in the number of nodes since approximately 2002 because there is an increase of both engineers and components. The tendency starts to decrease after approximately 2010 as other version control systems were adopted. Thus, after 2010, the data are a large sample rather than a comprehensive dataset. This tendency is also reflected in the number of edges of the bipartite graphs, which is correlated with the number of nodes [40]. The movement of some core technologies to a different versioning system is reinforced by the continuous increase in the number of connected components in the bipartite graph, which indicates a less integrated core of components.

Similarly, Figure 5 shows the time series of average degree, maximum degree, and maximum weight for the bipartite graphs. Note that under the presence of spikes in the measured signal, they are not correlated with the vertical lines that indicate the occurrence of the precipitating events. We revisit this issue in Section 4.2.

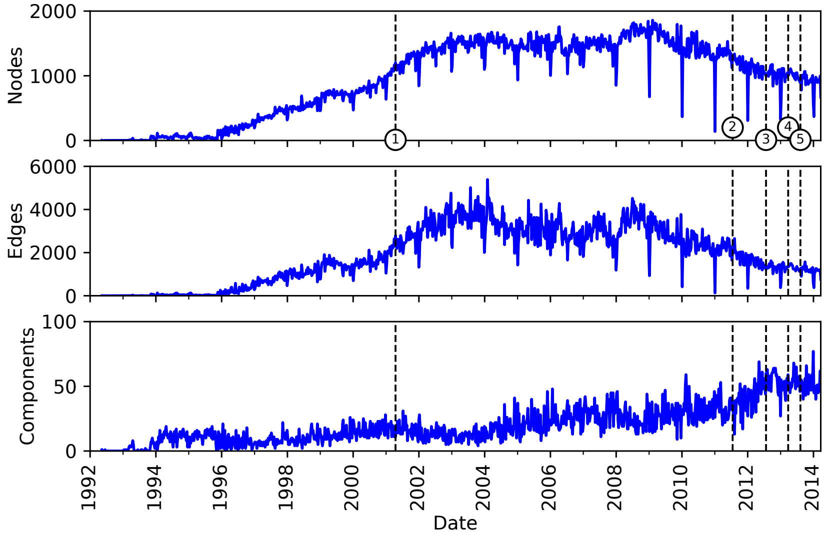

Figure 4: Time series of the number of nodes (top panel), edges (middle panel), and connected components (bottom panel) for the bipartite graphs.

4.2 One-mode projection graph properties series

Here, we report results on the same properties as we did for the bipartite graphs, i.e., nodes, edges, components, average degree, maximum degree, and maximum weight. Formalisms about the framework to build the graphs are defined in Section 3.3. Note that, in this case, edges occur when two engineers have interact with the same component (i.e., same code repository) based on the bipartite one-mode

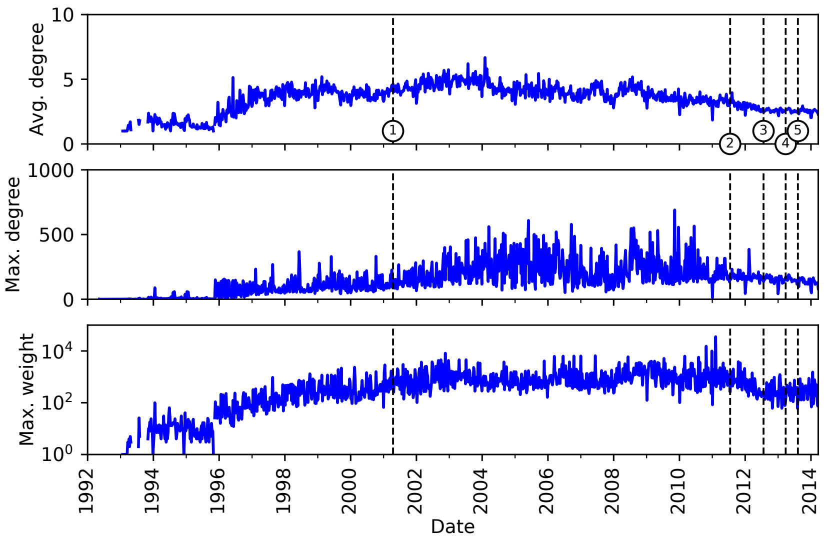

Figure 5: Time series of the avg. degree (top panel), max. degree (middle panel), and max. weight (bottom panel) for the bipartite graphs.

projection. The number of nodes is $|\mathscr{H}_\top(k)|$. This means the total number of engineers in the one-mode projection graph. The degree of node i is $d_i^\top(k)$, i.e., its number of neighbors. This means the number of other engineers connected to i. The set of edges of of the graph $\mathscr{G}_\top(k)$ is $\mathscr{E}_\top(k)$. The number of edges is $|\mathscr{E}_\top(k)| = \sum_{i\in\mathscr{V}_\top(k)} d_i^\top(k)/2$. This means the total number of interactions in the one-mode projection graph. A connected component is a subgraph in which any two nodes are connected to each other by paths. This meas subgraphs made of engineers in which any two vertices are connected, i.e., engineers that work in related components. The average degree of graph $\mathscr{G}_\top(k)$ is $2 \times |\mathscr{E}_\top(k)|/|\mathscr{H}_\top(k)|$. This means the average number of activity of engineers in the graph. The maximum degree of graph $\mathscr{G}_\top(k)$ is the maximum number of neighbors in the graph, i.e., $\max\{d_i(k), \forall i \in \mathscr{H}_\top(k)\}$. This means the degree of the node with more interactions. The maximum weight of a graph $\mathscr{G}_\top(k)$ is the maximum weight of edges in the graph, i.e., $\max\{\omega_{ij}(k), \forall i, j \in \mathscr{H}_\top(k)\}$. This means the weight of the interaction with maximum strength.

Figure 6 (top) shows the observed number of nodes (i.e., engineers in the user graph). Figure 6 (middle) shows the number of unique edges representing the number of interactions between engineers. Figure 6 (bottom) shows the number of connected components in the one-mode projection graphs. In general, for the number of nodes and edges, there is an increase in these measurements after roughly 2002. The tendency starts to decrease after approximately 2010 when other version control systems began to be adopted. Thus, after 2010, the data are a large sample rather than a comprehensive dataset. The movement of some core technologies to a different versioning system is reinforced by the continuous increase in the number of connected components in the one-mode projection graph, which indicates a less integrated core of components.

Similarly, Figure 7 shows the time series of average degree, maximum degree, and maximum weight respectively. Although there are several spikes for these measurements, we present an evaluation of the proposed algorithm, when these measurements inform the detection signature in Section 4.3.

4.3 Algorithm evaluation

We applied the proposed algorithm for anomaly event detection by leveraging on the structural properties of the one-mode projection graphs. Note that one-mode projection graphs are derived from the original

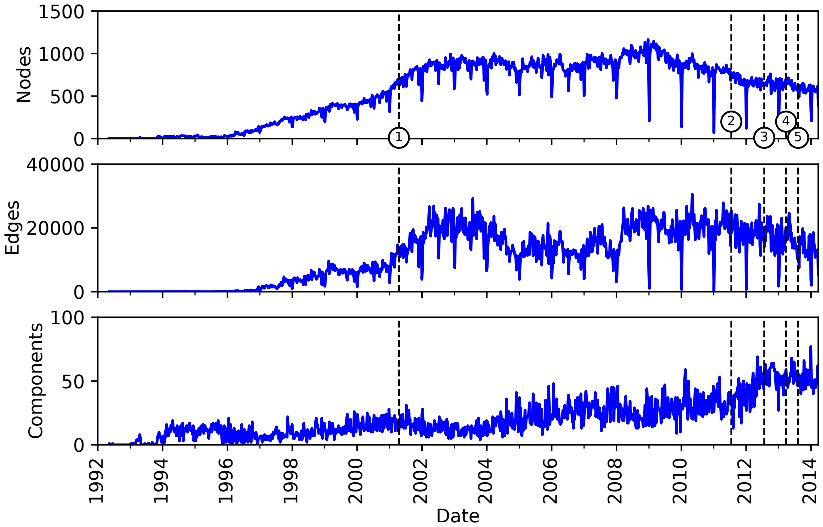

Figure 6: Time series of the number of nodes (top panel), edges (middle panel), and connected components (bottom panel) for the one-mode projection graphs.

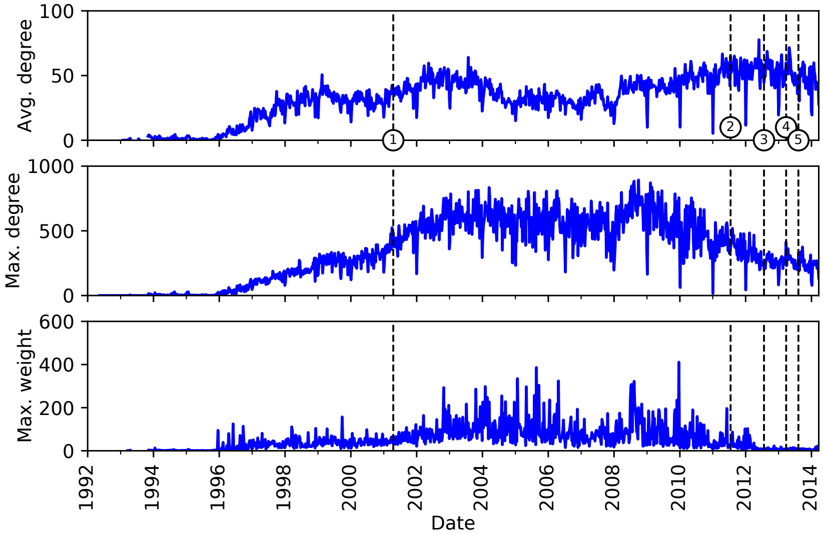

Figure 7: Time series of the avg. degree (top panel), max. degree (middle panel), and max. weight (bottom panel) for the one-mode projection graphs.

user-system interactions or bipartite graphs, so we compare the results of our proposed method with those properties that are measured in the equivalent one-mode projections. Our criteria for selection of anomalous time intervals is based on the idea of detecting observations that are far away from the median (for a specific time interval in which a one-mode projection graph is generated) as we specify in Algorithm 2. Following similar visualization conventions that we used in Section 4.2, in the following plots, the black horizontal line represents the median from the empirical observations. Each horizontal red band represents one standard deviation (the intensity of the bands is proportional to the distance with respect to the median). Remember that the standard deviation is estimated using the interquartile range of the distribution of these measurements. We estimated m_0, i.e., the length of the initial cumulative one-mode graph segment, by computing $\arg\max_{m_0} |C(\mathscr{G}_{\top}^{m_0})|$. That is achieved by the end of 2002, and it is the reason we report the following properties since January 1st, 2003.

Figure 8 shows the time series of nodes, edges, and connected components after the period of characterization of communities, i.e., the period of time comprehended between May 4, 1992 and December 31, 2002. As can be seen, even when there are some fluctuations in these measurements, the majority of the observations lay up to three standard deviations away from the median. This means that few time intervals were reported as anomalous during the observation period by relying in these properties.

We also performed similar experiments for the remaining graph-based properties, i.e., average degree, maximum degree, and maximum weight. In particular, Figure 9 summarizes these findings. For both average degree and maximum degree, the algorithm did not report suspicious time intervals given that the signal does not exceed ±3 standard deviations from the median. For the signal corresponding to the maximum weight, various spikes surprise the limits for detection. We report on the performance of these measurements later in Section 4.4.

Figure 10 shows the behavior for the proposed metric. Details on how this metric is derived are found in Equations 1 and 2. In particular, there are some spikes that exceed the threshold used by the algorithm and are close enough to the release of the precipitating events. These spikes suggest a drop in the number of edges between members of the same communities (conversely an increase in the number of edges between members of different communities) which, based on our proposal, means a diversified behavior, i.e., more interaction with different components.

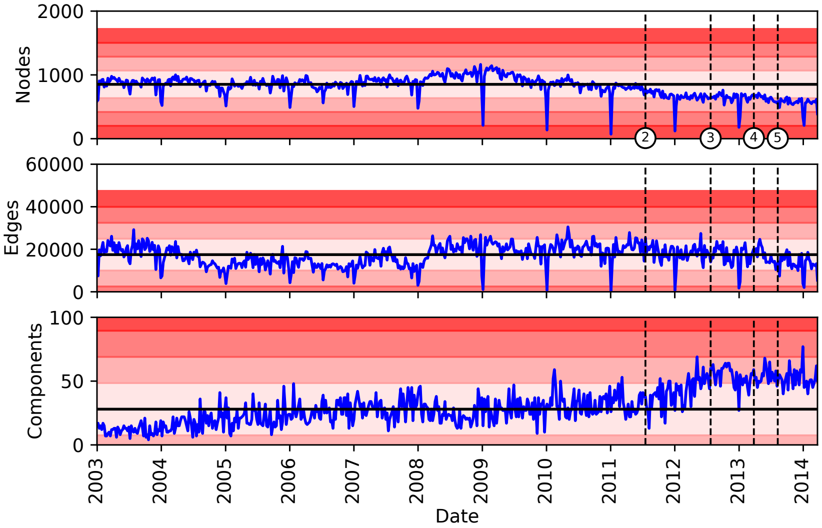

Figure 8: Time series of the number of nodes (top panel), edges (middle panel), and components for the one-mode projection graphs (bottom panel).

4.4 Algorithm performance

We compare the performance of the proposed algorithm with the performance of a random algorithm. In particular, let the output of the random algorithm be $\hat{R} = (\hat{R}_1, \ldots, \hat{R}_{\tilde{n}}) \overset{i.i.d.}{\sim} \text{Bernoulli}(0.5)$. This means that each time interval is equally likely to be selected as anomalous based on random chance.

Performance for all the proposed algorithms is compared based on accuracy, precision, recall, and F1 score. These measurements were estimated using the TP, FP, FN, and TN derived from Algorithm 1.

Accuracy is the most basic measure of performance for classification. It quantifies the proportion of correctly predicted positive and negative instances (i.e., time intervals classified as anomalous or not that were correctly classified). It is quantified as accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$.

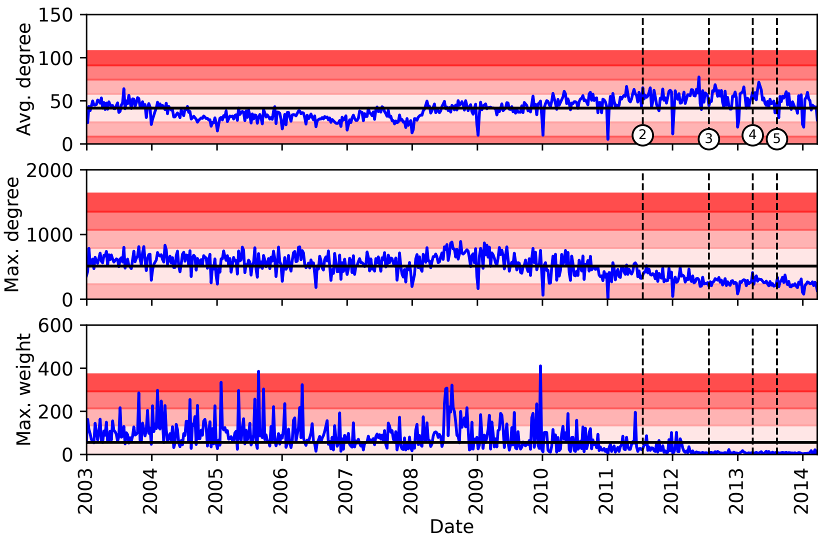

Figure 9: Time series of the avg. degree (top panel), max. degree (middle panel), and max. weight (bottom panel) for the one-mode projection graphs.

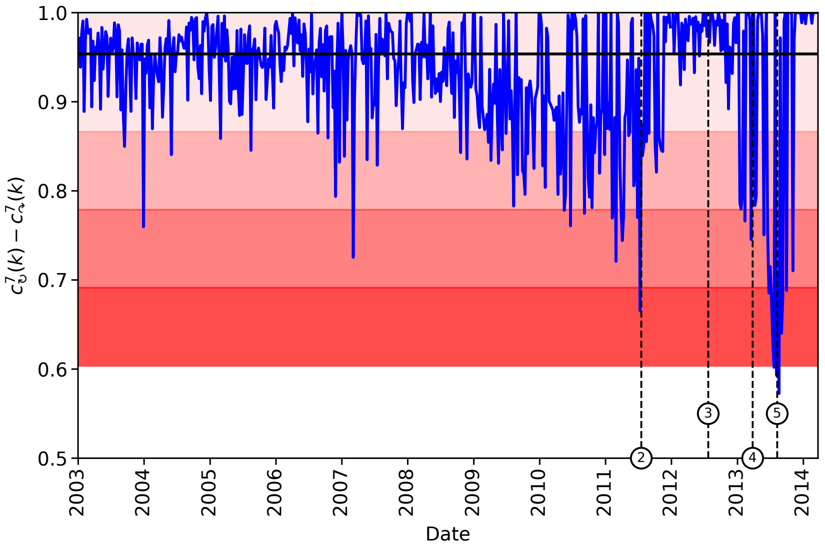

Figure 10: Time series of the intra- minus inter-community edge ratio for the one-mode projection graphs.

Precision quantifies the proportion of positive predictions that have been correctly classified. This means that if a considerable number of time intervals are erroneously classified as anomalous, then the algorithm has low precision. In other words, it is a measure of classification exactness. It is quantified as precision $= \frac{TP}{TP+FP}$.

Recall quantifies the proportion of actual anomalous intervals that have been predicted as positive. This means that if an insignificant number of time intervals are classified as anomalous but they are not, then the algorithm has low recall. In other words, it is a measure of classification completeness. It is quantified as recall $= \frac{TP}{TP+FN}$.

The F1 score conveys the balance between precision on and recall calculated through the harmonic mean. It is quantified as F1 score $= 2\frac{\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$.

Figures 11, 12, 13, 14 show the performance for different detection criteria, i.e., random, nodes, edges, connected components, average degree, maximum degree, maximum weight and the proposed approach under different detection resolutions. Performance in the random algorithm is calculated after $1,000$ realizations its evaluation. That means that for the random algorithm, we report on the mean and standard deviation on such measurements. As we might expect, the performance of the proposed approach starts increasing when the detection resolution is increased. For the maximum detection resolution that we used, i.e., $26m$, the results of the proposed approach outperforms the other measurements with a F1-score of approximately 85.7%. Noticeably, the performance of the random algorithm is even higher than those based on graph measurements even when taking into account the effect of the standard deviations represented by the error lines.

Accuracy of detection methods based on the graph-based properties is high given that the majority of time intervals are not marked as anomalous based on the small number of precipitating events (which makes this an unbalanced dataset).

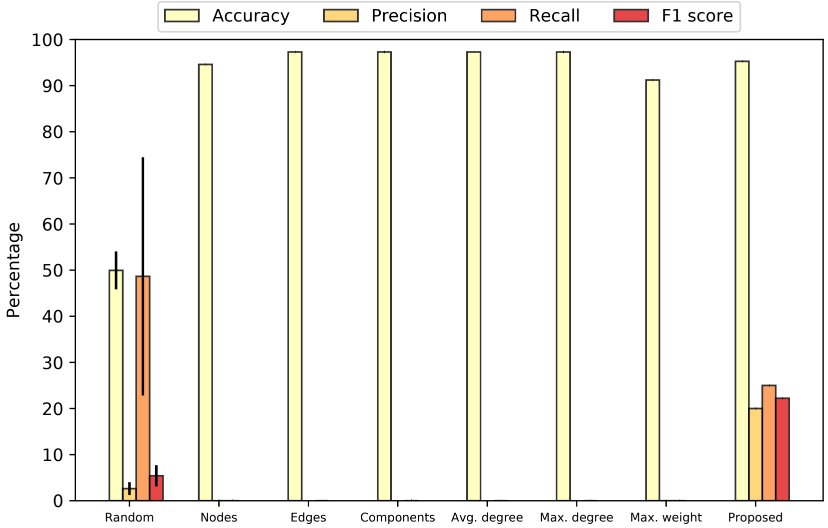

Figure 11: Algorithm performance for detection resolution $4m$.

5 Discussion

The main assumption behind the proposed approach is that insider threat events increase following certain types of events. Thus, counts of potentially malicious actions correlate with the announcement of precipitating events. Insider threat events are associated with stress or disgruntlement, and these responses are often triggered in the wake of precipitating events (e.g., [41, 42, 43, 44, 45]). This abstraction allows us to test the hypothesis as to whether the diversification of the committing behavior of users changes after the presence of a precipitating event.

We have proposed a bipartite graph framework that learns regular community behavior based on the interactions of engineers and components, and analyzes the patterns of connections in and between communities. We then use this to examine a time period that includes major precipitating events. As a result, the ground truth available for the analysis implemented here is the rate of insider risk in the organization after precipitating events. The validation of the model would be clear increases in the number of interactions across communities after precipitating events, and few increases without these.

Precision and recall together measure how often a threat is correctly identified and how often the non-malicious is correctly identified, i.e., no false positive or false negatives. This correctness is a significant

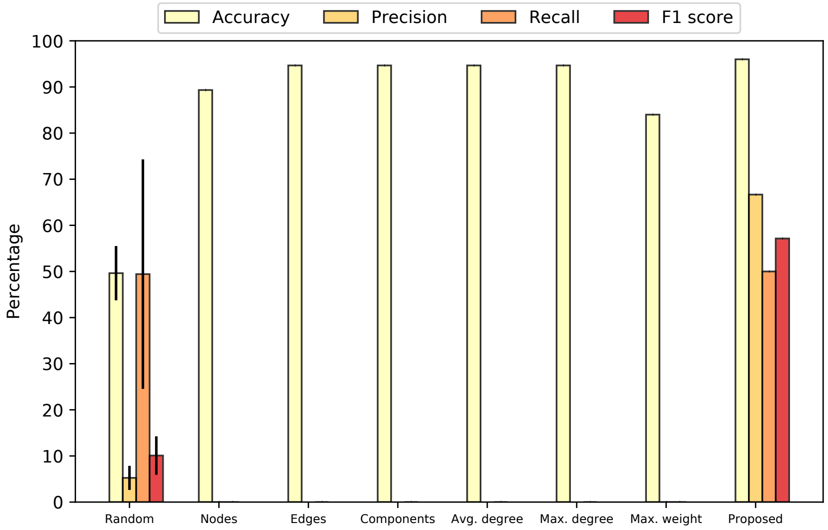

Figure 12: Algorithm performance for detection resolution 8*m*.

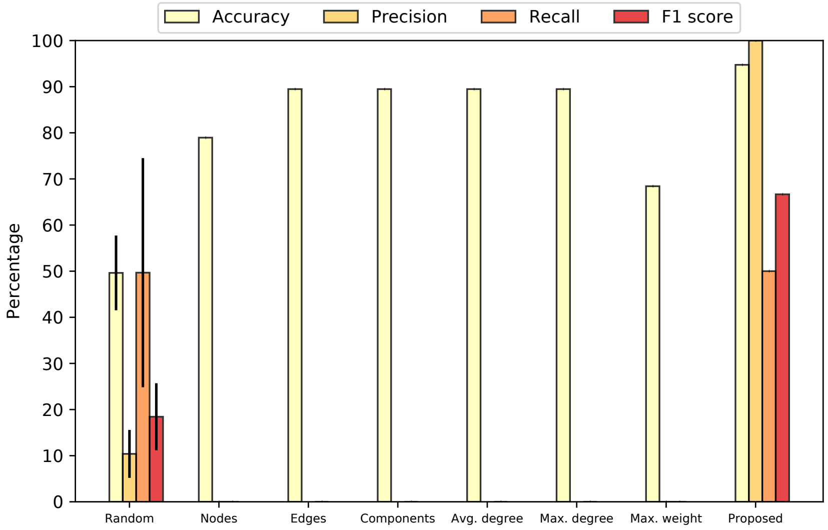

Figure 13: Algorithm performance for detection resolution 16*m*.

challenge in detecting insider threats. Individual organizational tolerance for false positives versus false negatives may differ. Figures 11–14 show that this trade-off can be changed by altering the detection resolution for the analysis.

Our approach makes a well-grounded assumption about the overall rate of insider threats and examines aggregate detection after precipitating events. Alternative approaches use artificial data with anomalies generated based on scenarios and confidential data. Another alternative is using qualitative research and directly leveraging known cases. By definition, the artificial data and case studies can only address the insider threats that have been detected using other methods. A third approach examines private datasets which includes potential malicious insider behavior. Our results use a private dataset subject and temporal analysis to illustrate that insider behavior increases are correlated with what are known to be precipitating events.

Of the three methods to address suspicious insider behavior, reproducibility is a particular strength of

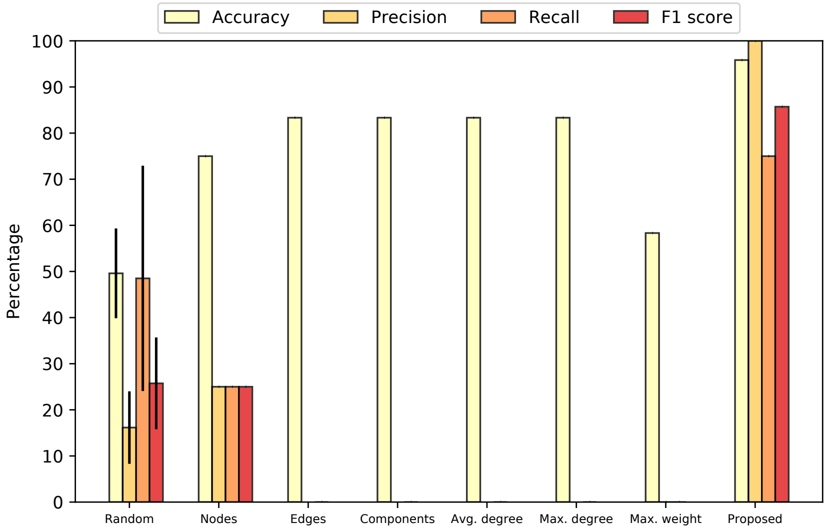

Figure 14: Algorithm performance for detection resolution 26*m*.

artificial data and is a particular challenge to the third approach (i.e., the one used here). The challenge to the second (case studies) and third (confidential data) approaches are of reproduction and validation. To address these challenges, we will release the scripts used to implement this model on or before publication of the paper. With the publication of our model as implemented, in addition to the description here, our analysis can be reproduced using any organization's private data. One goal in publishing this work is to encourage other researchers to use the model on the data available to them.

One requirement for this approach is adequate data to create the one-mode projection of the interactions between engineers from the bipartite graph of engineers and components. The current dataset covers more than two decades of interactions with engineers and version control systems. The requirements for the minimal training dataset is an open question. With logging provided by version control systems, software organizations have adequate data. However, other organizations with different types of data may struggle to find the optimal input. Another question is the optimal size of a community or subgraph [46]. This is a parameter that will vary between organizations.

One possible weakness to this approach is that an organization with a systematic insider threat problem may be unable to use this as detection. Training for community detection requires the insider's behavior to be anomalous. For example, organizations with high levels of turnover may consistently see behavior that would be anomalous in another organization, one with has higher retention or a more careful workforce.

Our approach identifies behaviors as opposed to focusing on the motivation of an individual. As a result, the particular strength of this method is identification of a significant number of suspicious behaviors across the entire employee population. A weakness is that an employee who becomes slowly malicious and increases suspicious behaviors over time may be able to train the model of that organization not to recognize his behavior as anomalous. This attack would be mitigated by the characterization of others in organization (who cannot be controlled by the insider). As with all insider threat detection systems, any employee who has access sufficient to manipulate the input and output of the model itself can defeat the analysis.

It might also be the case that our assumptions are incorrect. It may be the case also that precipitating events are not the only triggers to this type of activity. If insider threats are a result or response to specific events, other specific events including employee dismissal, dispute with employers, perceived injustices, family problems, coercion, or new opportunities—as has been highlighted in [20]—should be considered when evaluating the proposed approach.

6 Conclusions

In this paper, we have revisited the problem of insider threat event detection using graph mining analytics. To our knowledge, this is the first paper proposing an insider threat detection method using temporal bipartite graphs to pinpoint malicious events. Our main contribution is the proposal and evaluation of a generic analytical framework that builds on previous results in analysis of social networks to identify anomalous behavior by distinguishing access requests within and beyond a given community. We analyzed access to resources (i.e., code repositories) by employees (i.e., coders and engineers) using a time series of graph properties to pinpoint time intervals that identify suspicious insider behavior. The temporal analysis framework can be used with other datasets, including by organizations with no interest in sharing internal logs.

One major challenge in identification of potentially malicious behavior is determining ground truth. Although catastrophic insider events are well documented, the regular exfiltration of data by insiders is less well documented. There is a dearth of data. To address this, we examined the incidences of suspicious activity and correlated these with events known to be correlated with increases in insider threat behaviors, specifically precipitating events. The decision criteria for identifying these time intervals is based on quantifying changes in the way in which employees interact with resources after precipitating events have been announced. This performance analysis framework can be used by any organization that has experienced precipitating events in order to test it for applicability to its own risks. Further, by altering the time period for the analysis, organizations can make their own trade-offs as to the level of activity that will result in investigation.

From our results, it is possible to see that the proposed framework is able to identify time intervals in which anomalous activity happens with a reasonable F1 score. We compare the performance of the proposed approach with anomaly detection approaches based on a naive random and edge density dependent statistics. Our approach outperforms these intuitive approaches giving us insights on the importance of the diversification of committing behavior on user-system interactions as a possible indicator of insider threat.

There are three inter-related contributions of this work. The first is the use of temporal graph analysis to analyze insider threat actions as anomalies. The use of bipartite graphs is also an innovation. The second contribution is the performance analysis of this method as well as the empirical analysis. The third contribution is the use of graph decomposition to identify anomalous behaviors using measures of community structure (in addition to the widely-used volume metric). The final contribution is the manner in which the analytics leverage organizational theory insights on insiders and precipitating events. A particular strength of the approach is that all of these methods are reproducible by any organization with code or data repositories. We detail the technical contributions below.

- ***Temporal graph analysis framework**:* We propose a generic temporal graph analysis framework to model the evolution of bipartite graphs and their equivalent one-mode projection. The proposed framework is based on the idea that the evolution of user-system interactions can be abstracted as a set of consecutive graphs—also called graph stream (Section 3.2). This framework allows to select the granularity of network formation which has been found to be application dependent [47]. We use the proposed framework to formalize a set of measurements of the observed graphs at each time interval.

- ***Performance evaluation framework**:* We propose a generic framework to compute the performance of an event detector algorithm (Section 3.5). We compare the performance of the proposed method with a naive random and others edge dependent algorithms. (Section 4.4).

- ***Graph mining analytics:*** We use graph mining to reveal that some properties of the one-mode projection of the bipartite graph significantly change in the presence of precipitating events. To do this, we leverage more than 22 years of data on user-system interactions in a version control system. In particular, we show that users tend to diversify their patterns of interactions with components after a precipitating event is announced (Section 4.2). Our results suggest that this change in user behavior can be used to infer when anomalous events are happening before widespread disruption. Our work is differentiated from the work in [17] in three ways. First, we rely on the notion of community structure to inform the detection process. Second, we integrate the volume of interactions between users in different communities into the event detection. Finally, we quantify the perturbations inserted in the system after precipitating events that might lead to insider threats. Methodologically closest to our work is an analysis of the Enron email corpus and Twitter data in [48]. This work is differentiated not only by the domain (i.e., version control system) but also in that we abstract interactions as a bipartite graph and compare our detection results with standard detection approaches.

In summary, we abstract user-system interactions as a modeling framework and apply temporal graph analysis for identification of insider threat risks. We believe this approach could be widely applicable.

Future work is needed to explore the effect of the length of the initial cumulative graph segment (i.e., m_0) on the performance of the method for the current dataset.Future work also includes applying the method to other precipitating events (other than the limited restructuring events analyzed here) inside the same enterprise, to benchmark the performance of temporal bipartite community graphs against other event detection methods. We are currently seeking partnerships with other organizations to evaluate this graph mining approach using correlations with their distinct precipitating events. In addition to detecting aggregate behaviors during an organizational event, our temporal graph method has the potential to provide weighted estimates of risk for specific behaviors and specific individuals, both at the moment of the event and integrated over time. In the long run, our goal is the adoption of this as a mechanism to detect high-risk behaviors by insiders.

7 Acknowledgements

Pablo Moriano acknowledges Yong-Yeol Ahn for early feedback and discussions about the use of community diversification metrics for detecting anomalous events in Twitter data as well as Jorge Finke for his insights about the temporal abstraction for performance evaluation of detection algorithms. All authors thank Cisco ASIG members for help in collecting data and setting up experiments. This research was supported in part by the National Science Foundation under CNS-1565375, Cisco Research Support #591000, and the Comcast Innovation Fund. 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 the US Government, the National Science Foundation, Cisco, Comcast, or Indiana University.

References

[1] P. Moriano, J. Pendleton, S. Rich, and L. J. Camp, "Insider Threat Event Detection in User-System Interactions," in *Proc. of the 9th ACM CCS International Workshop on Managing Insider Security Threats (MIST'17), Dallas, Texas, USA*. ACM, October 2017, pp. 1–12.

[2] Reuters, "Ex-Ford engineer sentenced for trade secrets theft," http://www.reuters.com/article/us-djc-ford-tradesecrets-idUSTRE73C3FG20110413, [Online; Accessed on July 5, 2017], April 2011.

[3] FBI, "Fannie Mae Corporate Intruder Sentenced to Over Three Years in Prison for Attempting to Wipe Out Fannie Mae Financial Data ," https://archives.fbi.gov/archives/baltimore/press-releases/2010/ba121710.htm [Online; Accessed on July 5, 2017], December 2010.

[4] J. Edwards and M. Hoosenball, "NSA contractor charged with stealing secret data," http://www.reuters.com/article/us-usa-cybersecurity-arrest-idUSKCN12520Y, [Online; Accessed on July 5, 2017], October 2016.

[5] D. Culp, "Lessons not learned: Insider threats in pathogen research," http://thebulletin.org/lessons-not-learned-insider-threats-pathogen-research [Online; Accessed on March 28, 2018], April 2013.

[6] Ponemon Institute, "2016 Cost of Cyber Crime Study & the Risk of Business Innovation," Ponemon Institute, Tech. Rep., 2016.

[7] M. Bishop, H. M. Conboy, H. Phan, B. I. Simidchieva, G. S. Avrunin, L. A. Clarke, L. J. Osterweil, and S. Peisert, "Insider Threat Identification by Process Analysis," in *Proc. of the 2014 IEEE Security and Privacy Workshops (SPW'14), San Jose, California, USA*. IEEE, May 2014, pp. 251–264.

[8] A. Vespignani, "Predicting the behavior of techno-social systems," *Science*, vol. 325, no. 5939, pp. 425–428, July 2009.

[9] M. E. J. Newman, *Networks: An introduction*, 1st ed. Oxford University Press, 2010.

[10] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 626–688, May 2015.

[11] P. Holme and J. Saramäki, "Temporal networks," *Physics Reports*, vol. 519, no. 3, pp. 97–125, October 2012.

[12] S. Ranshous, S. Shen, D. K. S. Harenberg, C. Faloutsos, and N. F. Samatova, "Anomaly detection in dynamic networks: a survey," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 7, no. 3, pp. 223–247, May 2015.

[13] P. Parveen, J. Evans, B. Thuraisingham, K. W. Hamlen, and L. Khan, "Insider Threat Detection Using Stream Mining and Graph Mining," in *Proc. of the 3rd IEEE International Conference on Privacy, Security, Risk and Trust and the 3rd IEEE Inernational Conference on Social Computing, Boston, Massachusetts, USA*. IEEE, October 2011, pp. 1102–1110.

[14] W. Eberle, J. Graves, and L. Holder, "Insider Threat Detection Using a Graph-Based Approach," *Journal of Applied Security Research*, vol. 6, no. 1, pp. 32–81, December 2010.

[15] A. P. Moore, D. M. Cappelli, and R. F. Trzeciak, *The "Big Picture" of Insider IT Sabotage Across U.S. Critical Infrastructures*. Springer US, 2008, pp. 17–52.

[16] I. Barnett and J.-P. Onnela, "Change point detection in correlation networks," *Scientific reports*, vol. 6, p. 18893, January 2016.

[17] S. Heymann and B. Le Grand, "Monitoring user-system interactions through graph-based intrinsic dynamics analysis," in *Proc. of the 7th IEEE International Conference on Research Challenges in Information Science (RCIS'13), Paris, France*. IEEE, May 2013, pp. 1–10.

[18] T. Zhou, J. Ren, M. Medo, and Y.-C. Zhang, "Bipartite network projection and personal recommendation," *Physical Review E*, vol. 76, no. 4, p. 046115, October 2007.

[19] T. Rashid, I. Agrafiotis, and J. R. C. Nurse, "A new take on detecting insider threats: Exploring the use of hidden markov models," in *Proc. of the 8th ACM CCS International Workshop on Managing Insider Security Threats (MIST'16), Vienna, Austria*. ACM, October 2016, pp. 47–56.

[20] J. R. C. Nurse, O. Buckley, P. A. Legg, M. Goldsmith, S. Creese, G. R. T. Wright, and M. Whitty, "Understanding Insider Threat: A Framework for Characterising Attacks," in *Proc. of the 2014 IEEE Security and Privacy Workshops (SPW'14), San Jose, California, USA*. IEEE, May 2014, pp. 214–228.

[21] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Physics Reports*, vol. 659, pp. 1–44, November 2016.

[22] D. M. Cappelli, A. P. Moore, and R. F. Trzeciak, *The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes (Theft, Sabotage, Fraud)*, 1st ed. Addison-Wesley Professional, January 2012.

[23] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, September 1978.

[24] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu, "Graphscope: parameter-free mining of large time-evolving graphs," in *Proc. of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07), San Jose, California, USA*. ACM, August 2007, pp. 687–696.

[25] L. Akoglu and C. Faloutsos, "Event detection in time series of mobile communication graphs," in *Proc. of*

the 27th Army Science Conference, Orlando, Florida, USA, December 2010, pp. 77–79.

[26] D. Koutra, T.-Y. Ke, U. Kang, D. H. Chau, H.-K. K. Pao, and C. Faloutsos, "Unifying guilt-by-association approaches: Theorems and fast algorithms," in *Proc. of the 2011 Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD'11), Athens, Greece*, ser. Lecture Notes in Computer Science, vol. 6912, September 2011, pp. 245–260.

[27] C. C. Aggarwal, Y. Zhao, and P. Yu, "Outlier Detection in Graph Streams," in *Proc. of the 27th IEEE International Conference on Data Engineering (ICDE'11), Hannover, Germany*. IEEE, April 2011, pp. 399–409.

[28] D. Duan, Y. Li, Y. Jin, and Z. Lu, "Community mining on dynamic weighted directed graphs," in *Proc. of the 1st ACM International Workshop on Complex Networks Meet Information and Knowledge Management (CNIKM'09), Hong Kong, China*. ACM, November 2009, pp. 11–18.

[29] C. C. Noble and D. J. Cook, "Graph-based anomaly detection," in *Proc. of the 9th ACM SIG IKDDnternational Conference on Knowledge Discovery and Data Mining (KDD'03), Washington, D.C., USA*. ACM, August 2003, pp. 631–636.

[30] W. Eberle and L. Holder, "Scalable anomaly detection in graphs," *Intelligent Data Analysis*, vol. 19, no. 1, pp. 57–74, January 2015.

[31] A. D. Kent, L. M. Liebrock, and J. C. Neil, "Authentication graphs: Analyzing user behavior within an enterprise network," *Computers & Security*, vol. 48, pp. 150–166, February 2015.

[32] Y. Chen, S. Nyemba, W. Zhang, and B. Malin, "Specializing network analysis to detect anomalous insider actions," *Security Informatics*, vol. 1, no. 1, p. 5, December 2012.

[33] G. Jean-Loup and M. Latapy, "Bipartite Structure of All Complex Networks," *Information Processing Letters*, vol. 90, no. 5, pp. 215–221, June 2004.

[34] M. E. J. Newman, "Detecting community structure in networks," *The European Physical Journal B*, vol. 38, no. 2, pp. 321–330, March 2004.

[35] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, January 2008.

[36] L. Weng, F. Menczer, and Y.-Y. Ahn, "Virality prediction and community structure in social networks," *Scientific reports*, vol. 3, no. 1, p. 2522, August 2013.

[37] D. Koutra, J. Vogelstein, and C. Faloutsos, "DeltaCon: A Principled Massive-Graph Similarity Function," in *Proc. of the 13th SIAM International Conference on Data Mining (SDM'13), Austin, Texas, USA*, May 2013, pp. 162–170.

[38] iDatalabs, "Companies using IBM Rational ClearCase," https://idatalabs.com/tech/products/ibm-rational-clearcase, [Online; Accessed June 28, 2017], June 2017.

[39] L. Benamara and C. Magnien, "Estimating properties in dynamic systems: The case of churn in p2p networks," in *Proc. of the 2010 INFOCOM IEEE Conference on Computer Communications Workshops, San Diego, California, USA*. IEEE, March 2010, pp. 1–6.

[40] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations," in *Proc. of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, Chicago, Illinois, USA*. ACM, August 2005, pp. 177–187. [Online]. Available: http://doi.acm.org/10.1145/1081870.1081893

[41] S. Mishra and G. Dhillon, "Information Systems Security Governance Research: A Behavioral Perspective," in *Proc. of the 1st Annual Symposium on Information Assurance, Academic Track of the 9th Annual NYS Cyber Security Conference, Albany, New York, USA*, June 2006, pp. 27–35.

[42] M. Warkentin and R. Willison, "Behavioral and policy issues in information systems security: the insider threat," *European Journal of Information Systems*, vol. 18, no. 2, pp. 101–105, April 2009.

[43] F. L. Greitzer, P. Paulson, L. Kangas, L. R. Franklin, T. W. Edgar, and D. A. Frincke, "Predictive Modelling for Insider Threat Mitigation," Pacific Northwest National Laboratory, Tech. Rep. PNNL-65204, March 2009.

[44] A. P. Moore, D. A. Mundie, and M. L. Collins, "A System Dynamics Model for Investigating Early Detection of Insider Threat Risk," in *Proc. of the 31st International Conference of the System Dynamics Society, Cambridge, Maryland, USA*, July 2013.

[45] T. Benjaminsen, "The Norwegian Downsizing Approach in Terms of the Insider Threat-An interpretive

study," Master's thesis, Norwegian University of Science and Technology, 2017.

[46] Z. Dong, V. Garg, L. J. Camp, and A. Kapadia, "Pools, clubs and security: designing for a party not a person," in *Proc. of the 2012 New Security Paradigms Workshop (NSPW'12), Bertinoro, Italy*. ACM, September 2012, pp. 77–86.

[47] G. Krings, M. Karsai, S. Bernhardsson, V. D. Blondel, and J. Saramäki, "Effects of time window size and placement on the structure of an aggregated communication network," *EPJ Data Science*, vol. 1, no. 1, p. 4, May 2012.

[48] P. Moriano, J. Finke, and Y.-Y. Ahn, "Community-based anomalous event detection in temporal networks," in *Proc. of the 2017 Conference on Complex Systems (CCS'17), Cancún, Mexico*, September 2017.

Author Biography

Pablo Moriano received his B.S. and M.S. degrees in Electrical Engineering from Pontifica Universidad Javeriana, Cali, Colombia, in 2008 and 2011, respectively. He is currently pursuing his doctorate in in the School of Informatics, Computing, and Engineering with the Informatics department, where his research combines complex systems and security. He obtained a M.S. in Informatics in 2017. His research focuses on the development of analytical and data-driven models to detect and mitigate anomalies in complex systems. Applications range across multiple disciplines including Internet routing, prediction of catastrophic events in social media, and insider threat modeling in user-system interactions. He is a student member of IEEE and ACM.

Jared Pendleton has been a security researcher with Cisco's Advanced Security Initiatives Group (ASIG) for over 10 years, focusing on security audits and penetration testing of Cisco products. In recent years, Jared has been a member of ASIG's Forensics and Reversing Team, which specializes in analyzing threats to Cisco equipment, processes, and interests.

Steven Rich is a Principal Engineer in the ASRG group at Cisco and has been focusing on security research for the last 12 years. His main areas of interest are in the design and analysis of operating systems, networks, and general-purpose and domain-specific languages as well as how existing codebases and systems can be analyzed and hardened.

L. Jean Camp is a Professor at the School of Informatics, Computing, and Engineering at Indiana University. She is a Fellow of the Institute of Electrical and Electronic Engineers. She is a Fellow of the American Association for the Advancement of Science. She joined Indiana after eight years at Harvard's Kennedy School where her courses were also listed in Harvard Law, Harvard Business, and the Engineering Systems Division of MIT. She spent the year after earning her doctorate from Carnegie Mellon as a Senior Member of the Technical Staff at Sandia National Laboratories. She began her career as an engineer at Catawba Nuclear Station and with a MSEE at University of North Carolina at Charlotte. Her research focuses on the intersection of human and technical trust, levering economic models and human-centered design to create safe, secure systems.