Journal of Network and Systems Management (2019) 27:600–624
https://doi.org/10.1007/s10922-018-9478-8

Dynamic Link Anomaly Analysis for Network Security Management

Tao Zhang[1] · **Qi Liao**[1]

Received: 4 August 2016 / Revised: 25 August 2018 / Accepted: 7 November 2018 /
Published online: 13 November 2018

Abstract

Network management is challenging due to ever increasing complexity and dynamics of network interactions. While many changes in networks are normal, some changes are not. One of the daily tasks of network administrators is to identify and analyze these abnormal changes that are hard to find by traditional security mechanisms (IDS, firewall, anti-virus, etc.). This research conducts dynamic network analysis (DNA) and presents practical methodologies of data stream mining based dynamic link anomaly analysis (DLAA) using novel sliding time window structures and network analytics metrics. DLAA employs spatiotemporal link analysis to detect anomalies from dynamic network graphs. We formally define the network link anomaly types and use key link-structure similarity metrics and time-weighted functions to model the dynamics of topological changes. The methodology is generic in that it does not require additional information from nodes or links but only the topology itself. The DLAA framework consists of three algorithmic components including sliding time window, link scoring and link anomaly detection algorithms. Through experimental study on publicly available dataset, we demonstrate that the proposed DLAA framework has the capability to construct effective knowledge structures for measuring the security levels of large scale dynamic networks, and to provide insight for generalized DNA in network security domain.

Keywords Dynamic network analysis · Link anomaly · Network security management · Graph mining

✉ Qi Liao
liao1q@cmich.edu

Tao Zhang
taozhangncut@gmail.com

[1] Department of Computer Science, Central Michigan University, Mount Pleasant, MI 48859, USA

1 Introduction

Networks are everywhere, from enterprise computer networks to cloud computing infrastructures, from the Internet of computers to the Internet of Things, and from social networks to biological networks. The analytic complexity of networks has been increasingly challenging over the past decades due to the dynamics of network interactions and ever growing size of networks known as the big data era. Large-scale networks could suffer from threats from all aspects. For example, network anomalies such as failure events of IP backbone, denial of service (DoS) attacks, worm propagation, and network equipment outages, have distinct characteristics but are all potential threats, which may strongly affect the functionality and dependability of networks in varying degrees.

Dynamic network analysis (DNA) [1] is an emergent scientific field in network science. There has been an increasing interest in data stream mining based DNA for network security. The major challenge of this dynamic network analysis is that it is computationally expensive to separate anomalous changes from normal changes due to the constant structure changes in massive connectivity in a short amount of time. These anomalies are important and may be related to faulty hardware/software, misconfiguration, or security related events caused by malicious users and applications. Anomaly analysis is extremely useful in many domains. For example, network managers and administrators need to monitor the latest traffic graphs to increase situation awareness for both effective troubleshooting and time-efficient security-related investigation. While there has been effort in anomaly-based intrusion detection [2, 3], the anomaly analysis of dynamic network *links* has remained challenging.

In this paper, we study a branch of DNA, i.e., dynamic link anomaly analysis (DLAA) that focuses on detection of connection anomalies based on the graph topological structures such as information from network flow data. For better defense against the increasing 0-day attack, we aim to utilize behavior or learning based approaches, in particular, link prediction algorithms to analyze network connections. The challenges lie on a few key characteristics of anomaly detection that are fundamentally different from link prediction tasks [4]. Although link prediction [5, 6] may predict whether a pair of nodes that have not been connected in the past will ever be connected sometime in the future, its focus does not usually consider pairs of nodes that have previously been connected. The fundamental deficiency of traditional link predictions in the security domain is that they do not consider the *dynamics* of network connections, e.g., the on/off patterns. It has been observed that networks are not only becoming much larger but much more heterogenous, complex and dynamic as well. Taking computer networks for example, the massive amount of traffic among the computing nodes is constantly changing, e.g., users and applications may come and go at any time, establishing and tearing down the connections.

The contribution of this paper is notably the development of DLAA framework by incorporating link anomaly scoring and link anomaly detection algorithms for dynamic network graphs. To keep the algorithms generic, our approach does not

need background information such as node attributes, but is based on the network topologies. To solve the spatio-temporal link analysis challenges, we build our approaches on the dynamic graph structures by including a time dynamic function and similarity measurement based on the evolving network topologies in consecutive time windows. Intuitively, the frequency of links' appearances implies the importance of such connections. It is reasonable to assume the connections close to the time of investigation may receive more emphasis than connections happened much earlier on due to the temporal locality of packet exchanges and network flows. We further use the topological structures as similarity measurement such as Jaccard coefficients and Katz measure, and merge them into one coherent link anomaly methodology.

Different types of anomalies are formally modeled by considering every possible combination of previously unlinked/linked nodes and currently unlinked/linked nodes. In each time window, each pairwise nodes are assigned a normalized similarity score according to the aforementioned metrics and functions to measure their importance. The scores will be used to help determine whether the connections will be built or torn down in next time phase for a variety of situations. Based on the actual connectivity in a current snapshot graph, we are able to judge whether each link is anomalous or normal from the differences between the expected result and the reality. Through a case study and performance evaluation on publicly available dataset, we illustrate the effectiveness of the DLAA by comparing the accuracies, true and false positives and negatives. We believe the study has immediate benefits for network management in terms of security investigation and troubleshooting, and the methodologies are also general enough to have potential impact on many other types of networks.

The remaining of the paper is organized as follows. Section 2 discusses related work and compares proposed algorithms with exiting approaches. Section 3 defines the similarity metrics and anomaly types, and describes the key algorithms for dynamic link anomaly analysis. We then evaluate the effectiveness of the proposed dynamic link anomaly analytic methods through both extensive performance tradeoff study (Sect. 4) and case study (Sect. 5) using metrics such as ROC curves, accuracy and error rates, sensitivity, specificity and precision for various combinations of true/false positive/negative rates. Finally, Sect. 6 concludes the work.

2 Related Work

In areas of security management, network intrusion and anomaly detection [2, 7] can be roughly categorized as signature-based and data mining-based. While signature-based schemes have the advantage of low false positives, signatures require well defined patterns in advance, making the detection of 0-day exploits impossible as well as less effective for encrypted activities or self-modifying worms. There has been progress in using network behavior anomaly detection (NBAD) [8] to detect anomalies that are not pre-identified or not able to extract a clear pattern. As an integral part of network behavior analysis, NBAD still needs to establish the knowledge of normal network or user behavior over a certain period of time. Recent trend

on deep learning-based network anomaly detection via neural networks has shown promising results [9]. While machine learning based anomaly detection is promising since pre-defined signatures are not required, challenges remaining to researchers are the lack of attack-free clean training data, difficulty of evaluation, and high false rates [10].

Networked data has been historically represented in network graph format, and anomaly detection in graphs has gained awareness among researchers. For example, anomalies can be manually identified in network flow data by applying the subspace method using multivariate time series of byte counts, packet counts and IP-flow counts [7]. Neighborhood formation has been studied for bipartite graphs [11]. OddBall [12] detects anomaly in graph by assigning an "outlierness" score to each node. These research studies focus on neighborhood, or sphere, around each node, or the ego subgraph. Various distributions such as power laws in density, weights, ranks and eigenvalues of these neighborhood sub-graphs have been proved useful for anomaly detection. Graph-theoretic approach [13] has also been applied to detect anomaly in email networks from the publicly available Enron email corpus. In addition, both link structures and semantics (attributes) have been used to detect outliers in complex networks [14].

Although there has been significantly amount of work on anomaly detection [3] looking at various aspects of network anomalies (e.g., traffic volume variations, packet/flow classifications, DoS/DDoS attacks, etc), we note that the different focus of this work is on detecting link anomalies in dynamic graphs that exhibit on/off patterns. We complement existing work in the network security management domain by investigating harder questions, e.g., given only two graphs without any other knowledge, can we tell possible anomalies (4 types of edge anomalies as illustrated in Fig. 2)? The proposed algorithm is also generic that is based only on network topologies and is not restricted to IP networks.

Dynamic network analysis (DNA) [1, 15] aims to understand the complex and dynamic features of modern networks. DNA has profound implications for many important tasks. For example, a network manager may need to gain situational awareness and detect anomalies in information technology network for potential faults and security breach. DNA with network metrics for centrality and clustering can be applied to engineering design projects consisting of networks of people and activities [16].

Dynamic network analysis is highly challenging due to spatial–temporal network dynamics in terms of both topological structure and attribute evolution. Anomaly detection approaches in dynamic networks can be either distance/similarity-based or community-based. Community detection in both static and dynamic networks is helpful to detect network anomalies [17]. A statistical infinite feature cascade approach is proposed to detect link anomaly in dynamic social networks [18]. In addition, the evolution of relationships between entities in DNA can be understood through dynamic graph visualization [19].

Among the network metrics used, the Jaccard index, which is originally designed to measure the similarity between sample sets, can be useful to measure nodes' community similarity. The Katz index [20] is useful in measuring the importance of nodes (or centrality). Both measures have seen applications in social network

analysis. Jaccard based scores, while efficient to compute, may be restricted to local neighborhood only. Calculating the Katz measure may be slow if designed improperly as the graph size reaches up to a certain extent due to its cubic complexity. We combine these measures and time factor to compute one overall score for link anomaly analysis in dynamic networks.

In data mining domains, link mining [21, 22] and particularly link prediction algorithms [4, 23] are introduced as one of the popular graph mining areas. Researches summarize recent progress in link prediction and show its applications for a variety of domains such as social networks [5, 6], co-authorship networks [24], healthcare and bioinformatics [25]. However, most works in link prediction are often interested in predicting whether a pair of nodes that are not connected previously will ever be connected in the future. Although the link prediction algorithms will predict the feature connections in the graph, most of them use a static graph as learning set without considering the temporal information inherent in the traffic data. In computer security research, the network is highly time dynamic [26] and traditional link prediction hardly works well on these datasets with spatiotemporal dynamics. Instead, we focus on the task of link anomalies [11–13, 27–29] and address more the *dynamic* nature of link anomaly with "on/off" behaviors, e.g., whether and when a previously connected link will become disconnected, and vice versa.

3 Dynamic Link Anomaly Analysis

In this section, we begin by discussing how graphs may be derived from network traffic log by considering the source and destination IP addresses and port numbers. We then introduce the graph node similarity metrics and time dynamic functions for measuring the spatiotemporal dynamics of network. After describing how the system works and all possible situations for link anomaly predictions, we discuss the core link anomaly detection algorithms.

3.1 Graph Construction

The system processes the Cisco netflow-like data, which contains important information such as source and destination IP addresses, port numbers, start and end timestamps, protocols, etc. We primarily focus on two types of graphs, i.e., IP graphs and IP-port graphs. In IP graphs, each node represents one IP address and the edge represents the network connection between the source and destination IP pair. The IP-port graphs further consider the application protocols by introducing the port numbers, e.g., srcIP → srcPort → dstPort → dstIP is one possible way to construct the IP-port graphs, resulting in heterogenous graphs. Additional parameters such as ports may be useful in settings such as traffic flow classification [30].

For any moment, we have two types of graphs for anomaly detection: the prior graph and the current graph. The prior graph is constructed by combining all network connections from netflow data during the past, and will be updated for each sliding time window. The prior graph is the one that our link anomaly algorithms

can use to learn from the past and make educated guess on future connections. Since we do not have the luxury to do supervised learning, i.e., no one can ever mark each connection as either good or bad, the unsupervised learning is more suitable for learning the normal connection behaviors directly from the prior graph. It is therefore challenging that in real world situation, the prior graph may contain both clean data and attack data. It might be useful to freeze or adjust the models once anomaly has been confirmed to eliminate or minimize the data noise from the prior graph. In contrast to the prior graph, the current graph is built based on the network traffic in netflow data in the current time window. The current graph represents the most recent traffic reported by the netflow and is the one upon which our link anomaly detection algorithm makes a decision. Both the prior and current graphs will be rebuilt when the sliding time windows move, as illustrated in Fig. 1.

Real world networks are dynamic, i.e., constantly changing. Some changes are normal while others are not. To detect abnormal changes, we utilize spatiotemporal metrics to measure the topological changes in time-series snapshot graphs, as explained in the following sections.

3.2 Similarity Metrics

Let $G = (V, E)$ be the graph that represents the topological structure of a general connected network. Edges in the graph are denoted by $e = (u, v) \in E$, where $u, v \in V$, and V is the set of nodes or vertices. For each $u \in V, \Gamma(u)$ represents the set of u's distinct adjacent nodes (or neighbors). In addition, the set of paths between u and v is defined as $paths_{u,v}^{(l)}$ where l represents the exact length of the path. Each edge $e \in E$ counts 1 for unweighted graphs and $w(e)$ for weighted graphs, in which $w \in W$ is referred to as the related edge e's weight, and W is the set of edges' weights.

3.2.1 Jaccard's Coefficient

The Jaccard coefficient is well known to measure the similarity between sample sets, which is defined as the size of the intersection divided by the size of the union of the sample sets [31]. Particularly, in the graph G, for each pairwise nodes $v, u \in V$, the coefficient is defined as the ratio between the number of their common neighbors and the number of total neighbors, namely:

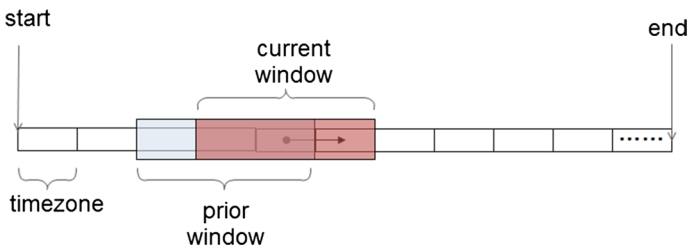


Fig. 1 The sliding time window will update and rebuild the prior graph and current graph of network connections

$$J(u, v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|} \tag{1}$$

In our approach, for instance, it indicates whether two network nodes, e.g., clients, servers, routers, have a large percentage of overlapping destinations regardless of the absolute number of connecting targets.

3.2.2 Katz Index

Originally proposed as a path-ensemble based proximity measure by Katz [20], the Katz measure is a variant of the shortest-path measure [31], which considers that the more simple paths there are between two nodes and the shorter these paths are, the stronger is the relationship of two nodes. The measure is defined as:

$$K(u, v) = \sum_{l=1}^{\infty} \beta^l \cdot \left| paths_{u,v}^{\langle l \rangle} \right| \tag{2}$$

where $0 < \beta < 1$ is a parameter ensuring that the shorter path contributes more to the score. However, the Katz measure has two main limitations when applied to real world networks. The first one is its cubic time complexity which makes it hard to be feasible for a large network. Since the shorter paths dominate the score, we set a maximum path length for larger graphs, i.e., only the paths shorter than the max value would count towards the Katz measure. The second limitation of the Katz measure is that it is inherently not geared towards the needs of some types of link anomaly detection. For example, the Katz measure for two nodes that are already connected could be nearly equal because the shortest path for them is only one hop away, which could dominate the score in most unweighted graphs. For the set of connected node pairs, utilizing Jaccard's coefficient may make more sense. Another challenge lies in heterogeneous graphs, such as those containing both IP addresses and port numbers, since some of pairwise nodes never have directly connected nodes as their common neighbors. Moreover, in some cases, only one snapshot graph is not enough for extracting normal nodes' connecting behaviors. Therefore, for the set of connected node pairs, we apply a time frequency function with the third temporal dimension and a sliding-window learning mechanism.

3.3 Time Frequency

We use a weighted time frequency function [32] to consider the time dynamics of connected nodes for link anomaly detection:

$$P(L_i) = \frac{\sum_{t=1}^{N} w(t) \cdot d}{\sum_{t=1}^{N} w(t)}, \quad d_{t,i} \in (0, 1) \tag{3}$$

$$w(t) = e^{-\lambda\left(1-\frac{t}{N}\right)} \tag{4}$$

The appearance probability functions can be either weighted or unweighted. The weighted form (Eq. 3) takes a nonlinear time weighting function $w(t)$ (Eq. 4), i.e., the appearance of links at later snapshot graphs (or in other words closer to the time of investigation) should have higher weights over the earlier graphs. Both Eqs. 3 and 4 are normalized between 0 and 1,where $P(L_i)$ represents the probability of ith link; $t = 1, 2, \ldots, N$; N denotes the total number of snapshot graphs; and $d_{t,i}$ takes a binary form to denote whether ith link appears or not at time t. It is reasonable to make such assumption due to the inherent temporal locality of network connections. Schemes such as network caching and Cisco netflow export also assume if a packet is observed between a source–destination IP/port pair, it is likely that the source will send packets again to the destination in the near future, thus later appearance of connections is more important (thus having higher weight) than earlier ones.

3.4 Link Anomaly Situations

Theoretically, for each pairwise connection, there could be as many as 16 possible situations of link anomaly prediction outcomes, as shown in Fig. 2. Each situation individually depends on its connection status in both the prior and current graphs as well as the verified results (i.e., whether it is really an anomaly or not). Figure 2 shows a decision-tree-like structure to illustrate the hierarchy of relationships regarding the 16 situations. There are four probabilities of connecting status between two vertices in the previous time slice and the current time slice. First, in the *build* section, a pair of nodes do not connect in the prior graph but are connected in the current graph, i.e., they build new connections. Second, in the *keep* section, two nodes are already connected in the prior graph and are still connected in the current graph, or in other words, they keep their existing connections. Third, in the *tear* section, a pair of nodes are connected in the prior graph but are disconnected in the current graph, i.e., they tear down the connections. Lastly, in the *never linked*

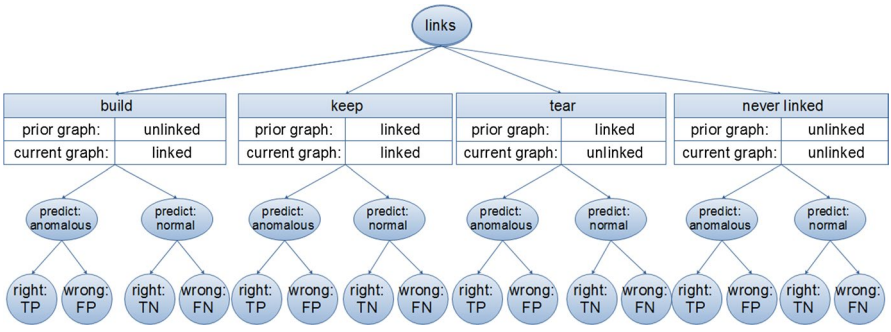


Fig. 2 The complete 16 situations of dynamic link anomaly analysis and detection

section, two nodes are never connected in either the prior or the current graph. For each of the above connection status, there could be two branches: either predicted anomalous or not anomalous. For each anomaly prediction, the prediction can turn out to be either correct or wrong, resulting in true positive (TP), false positive (FP), true negative (TN) and false negative (FN).

For instance, in a *build* scenario, a new connection made to a financial server from a workstation in the sales department, which has not connected to the server before, can be suspicious. It could indicate that a malicious user or an attacker is trying to compromise the machine and steal sensitive information. This type of link anomaly could be useful to detect security-related incidents such as port scans, DoS attacks, Trojan, worm and various malware infections.

Another example would be in the *tear* section, a DNS server that is previously connected by clients is no longer connected with a few machines. This could be indication of various issues possibly due to hardware faults, link failures, firewall issues, or simply misconfigurations. This type of link anomaly detection could be useful for fault localization, debugging and troubleshooting. Since both the malicious activities and equipment failures need at least one connection either in the last time slice or the current time slice, in this paper we analyze the first three scenarios while leaving the last *never linked* case for future study. In Sect. 4, we evaluate and discuss the various situations of link anomaly detection.

3.5 Link Anomaly Algorithms

In this section, we discuss our link anomaly algorithms via two core modules: the link anomaly scoring algorithm (Algorithm 1), and the link anomaly detection algorithm (Algorithm 2). The link anomaly scoring algorithm will calculate links' connecting scores based on the node similarity and weighted time frequency functions discussed earlier for various link anomaly situations, i.e., keep, build and tear. The link anomaly detection algorithm then generates the anomalous link list based on the scores using user-defined thresholds.

3.5.1 Link Anomaly Scoring Algorithm

Algorithm 1: Link Anomaly Scoring Algorithm $(\beta, \gamma, \gamma', G, G', I)$

Input: β — a parameter for the Katz measure
γ, γ' — weights for combination of two measures
$G = \langle V, E \rangle$ — a prior graph
$G' = \langle V', E' \rangle$ — a current graph
$I = \{I_i\}$ — raw data records, including timestamps, sources, destinations and operations
Output: $R = \{r_i\}$ — a table containing scores for pairwise nodes

```
1  begin
2      let R be a new table for storing scores for pairwise nodes
3      foreach (u, v) ∈ E do
4          r_{u,v} ← TimeFrequency(u, v, I)
5          j_{u,v} ← Jaccard(u, v, G)
6          R ← r_{u,v} · γ + j_{u,v} · (1 − γ)
7      end
8      foreach (u, v) ∈ E' do
9          if (u, v) ∉ E and u, v ∈ V then
10             k_{u,v} ← Katz(u, v, G, β)
11             j_{u,v} ← Jaccard(u, v, G)
12             R ← k_{u,v} · γ' + j_{u,v} · (1 − γ')
13     end
14     return R
15 end
```

According to different link anomaly situations (Sect. 3.4), we use node similarity functions (Eqs. 1 and 2) and weighted time frequency functions (Eqs. 3 and 4) separately to calculate pairwise node connection scores. For those links that have not been built yet in the past time slice but are connected at present, we use similarity-based algorithms. For links that have already been built in the past time slice, we select the time frequency algorithm, which predicts links' connecting possibility for the current graph. The algorithm extracts links' density on the basis of connections' frequency using build-and-tear patterns.

We first pick both a local similarity index and a global similarity index, i.e., the Jaccard Index and the unweighted Katz index. For each unlinked pair of nodes *u* and *v* in the prior graph, the Jaccard index value will reflect *u* and *v*'s local circumstance of neighborhoods. Link probability will be higher if *u* and *v* have more common neighbors, which carry more weights. We apply the Katz index to measure *u* and *v*'s linking possibility by considering the topological distance between *u* and *v* in the prior graph. Simply put, the assumption is nodes that are closer and have more paths

between them are more likely to connect than nodes that are remote. In particular, all simple paths between u and v, which could be limited by hops, participate in calculating the Katz index of u and v. Furthermore, all the paths $e_{u,v}$ and their lengths $w_{u,v}$ are used to predict the likelihood of direct connections between the two nodes. In order to solve the cubic complexity of the Katz measure and make the calculation more efficient for large networks, we apply a heuristic method of partial simple paths by using the maximum length as a limiting factor. Paths which are longer than the maximum value will not be counted, partly due to the fact that in most cases, paths of shorter lengths will dominate the measure.

For each linked pair of nodes u and v in the prior graph, weighted time frequency function is applied based on the temporal locality assumption in which connections made closer to the time of investigation will be more likely to reconnect than those connections happened at earlier time. As a result, the later-built links' scores will be much higher, suggesting a higher possibility of connection in the current graph. In addition, the Jaccard index may also be applied to linked nodes to measure their immediate neighborhood similarity. Intuitively, if two nodes have already been connected and have many common neighboring nodes, the chance for the two nodes to stay connected or to be connected again in the future is very high. To combine the scores from the above measures, weighted sums are adopted, as shown in Algorithm 1.

The computational complexity of the link anomaly scoring algorithm is polynomial time $O(n^3m)$, where $n = |V|$ is the number of nodes in the prior graph, $m = |E|$ is the number of network connections in the current graph. Since it is less efficient to compute scores for all pairwise nodes in the prior graph, only nodes that are actually connected in the current graph are used to compute the scores, thus m times. Relevant methods such as the Jaccard, Katz, and time frequency scores are applied for each link. Due to the cubic complexity time of the Katz measure which dominates the time complexity ($O(n^3)$), the overall complexity is $O(n^3m)$. If other faster algorithms, such as Fast Katz [33] that runs at $O(n + m)$, are used, the computational complexity could be further reduced to $O(m^2 + mn)$.

3.5.2 Link Anomaly Detection Algorithm

Algorithm 2: Link Anomaly Detection Algorithm (R, G, G', TH)

Input: $R = \{r_i\}$ — a table containing scores for pairwise nodes
$G = \langle V, E \rangle$ — a prior graph
$G' = \langle V', E' \rangle$ — a current graph
$TH = \{\theta_k, \theta_b, \theta_t\}$ — thresholds for different anomaly situations (keep, build and tear)
Output: A — a list of all anomalous links detected in the current time window

```
1  begin
2      let A be a new list
3      foreach e ∈ E do
4          if e ∈ E' and r_e < θ_k then
5              add e to the anomaly list A
6          else if e ∉ E' and r_e > θ_t then
7              add e to the anomaly list A
8      end
9      foreach  e ∈ E' do
10         if e ∉ E and r_e < θ_b then
11             add e to the anomaly list A
12     end
13     return A
14 end
```

Our prediction on whether a link in a network graph is anomalous or normal is based on the consistency between the link scores from the prior graph and the actual linkage in the current graph. For example, suppose a link existed in the prior graph and the link score is considerably high (over a specified threshold). If such a link is torn town in the current graph, that is in contrary to what is suggested by the scores. Therefore, we decide such a link disconnection is anomalous. For another example, suppose there is a connection in the current graph, however, the score of such a link, which may or may not exist in the prior graph, is quite low. Since the score is in contrary to the fact, we would consider the connection as anomalous because the two network elements are unlikely to be connected according to our prediction. Similarly, if the actual network connection matches our prediction, appear or not appear, based on the scores, then we consider these links as normal. The logic is summarized in Algorithm 2. For the link anomaly detection algorithm, the computational complexity is time $O(m)$ where m is the number of network connections.

While implementation specific, the score tables are divided into three segments, i.e., *keep* (c–c), *build* (d–c), and *tear* (c–d) tables. The *keep* table stores all connections that exist in both prior and current graphs. The *build* table stores connections

in the current graph but not in the prior graph. The *tear* table stores connections that exist in the prior graph but not in the current graph. The three thresholds ($\{\theta_k, \theta_b, \theta_t\}$ are for each table, respectively. Assuming the score tables are sorted by the score values (top = highest; bottom = lowest), the thresholds in both *keep* and *build* tables are on the bottom part meaning the connections have low scores but actually appeared. On the other hand, the threshold in the *tear* table is at the top meaning the connections have high scores but did not appear. Both are considered anomalies.

Note that the thresholds refer to the percentage of total links that will be predicted as anomaly. We use percentage thresholds in the score table to predict the appearance of links. Choosing the ideal thresholds is the common challenge for any system that involves a predefined threshold. The common way is to try and repeat until the ideal threshold is found that yields the most TP and TN. We will illustrate this task more in the case study and evaluation section. Another possible way is to use automatic statistical methods such as computing the mean (μ) and standard deviation (σ). For example, a score $S > \mu + \lambda\sigma$ will be predicted as appear, and vice versa. We note that there could be another way that is to use the simple K-means to perform clustering since the pairwise distances can be readily derived from the scores. In each way, we may eliminate the usage of thresholds, which could be our future work.

3.6 System Overview

Figure 3 shows the major components and the overall flow of the link anomaly detection system. The system will first process the raw data (netflow, firewall log, etc) and extract the connection information, i.e., timestamp, source and destination IP addresses (and optionally port numbers). This basic information will allow us to know who connects to whom and build the connectivity graph. The prior graph and the current graph will be constructed according to the sliding window controller (Fig. 1) with a customizable time window size. The process will keep repeating until the sliding window moves to the end of the dataset. In order to build connectivity graphs, a TCP connection state table is constructed from the flow data to keep track of all connection status. For example, even if a pair of source and destination nodes

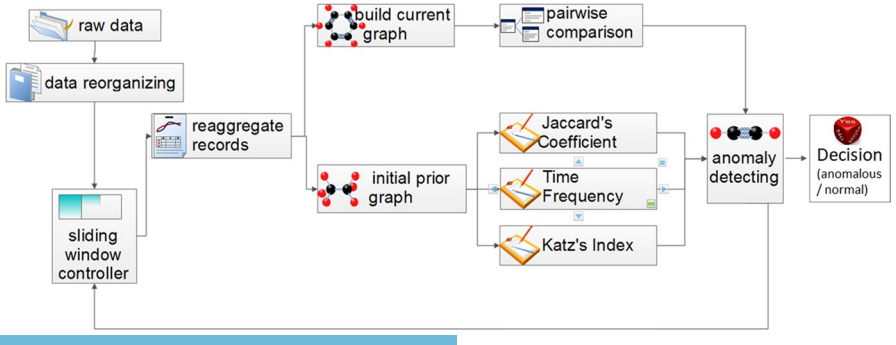


Fig. 3 Architecture overview defines components and overall process of link anomaly detection system

do not appear in the netflow record in the current time window, if that pair of nodes are still in the "connected" state, the connection should be included as an edge in the current graph, until an explicit "tear down" is read from the flow data, which then removes such a connection from the state table.

The prior graph will be used to calculate the link scores using a combination of the Jaccard's coefficient, Katz's index and weighted time frequency functions described in earlier sections, and store the results in a link anomaly score table. For each pairwise nodes in the current graph, lookups is performed in the score table. Based on the score and the actual linkage in the current graph, a decision of whether such a link is anomalous or normal is made. Finally, for verification purpose, an IDS log is checked to see if the link anomaly prediction matches the record. After finishing the current graph, sliding window controller will be called for computing a new time series of graphs, and the procedures will be repeated.

4 Tradeoff Study

For each of network connectivity situations in Fig. 2, there are two predicting results: anomalous (positive) or normal (negative). For each prediction, there can only be two results, true anomaly or false anomaly. From additional dataset such as IDS log, we are able to evaluate the link anomaly detection results for each connection situation via confusion matrix. For example, a correct prediction of an anomaly is a true positive (short as TP); a wrong prediction of an anomaly is a false positive (short as FP); a correct prediction of a normal connection is a true negative (short as TN); a wrong prediction of a normal connection counts as a false negative (short as FN). In other words, TP means we successfully identify the real anomaly connections which are indeed caused by malicious attacks, and TN refers to the negative (normal) tuples that are correctly labeled by the link anomaly detection system. FP means a link is identified as anomaly (positive) but it turns out to be normal (negative). FN means a link is considered as normal (negative) but turns out to be bad (positive).

The above four measures are used as the building blocks for our tradeoff comparative study. We use widely recognized metrics (see Table 1) for evaluating the link anomaly detection algorithm. These metrics include accuracy, error rate, sensitivity, specificity and precision. To begin with, accuracy (Eq. 5), also known as recognition rate, is the percentage of connections (normal and abnormal) that are correctly identified by our link anomaly detection method. Error rate (Eq. 6), on the other hand, measures the overall percentage of incorrect predictions of connections (normal and abnormal). Sensitivity (Eq. 7) is also referred to as the true positive rate (TPR), while specificity (Eq. 8) is also referred to as the true negative rate (TNR). The former is the proportion of anomalies that are correctly detected, and the latter is the proportion of normal links that are correctly identified. Precision (Eq. 9), also known as positive predictive value, is the proportion of detected anomalies that turn out to be true anomalies over all detected anomalies. Note it looks at a different aspect than the true positive rate (TPR) or sensitivity.

Table 1 Evaluation metrics

Metric	Equation	
Accuracy	$\frac{TP+TN}{P+N}$	(5)
Error rate	$\frac{FP+FN}{P+N}$	(6)
Sensitivity	$\frac{TP}{TP+FN}$	(7)
Specificity	$\frac{TN}{TN+FP}$	(8)
Precision	$\frac{TP}{TP+FP}$	(9)

In this section and Sect. 5, we use VAST Challenge'11 (MC2) dataset for illustration purpose. The dataset consists of a scenario for a large shipping company and 3 days of firewall and IDS logs as core information data from the corporate network.

We first create the receiver operating characteristic (ROC) curves in Fig. 4 for illustration. ROC curves are commonly used to illustrate the performance of a binary classifier with varying thresholds. We use ROC curves to plot TPR versus FPR at various threshold settings. We apply the thresholds and prediction methods as the link anomaly detection scheme. In addition, a random scheme is added

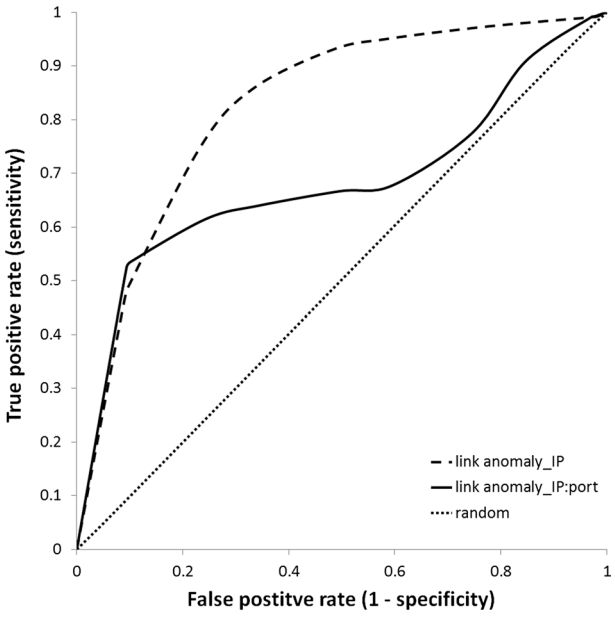


Fig. 4 TPR–FPR ROC curves for the link anomaly detection algorithm using IP graphs and IP-port graphs. The higher and quicker the curve rises, the better is the system

for comparison (the diagonal line) which is simulated by assigning links with random scores from 0 to 1 to form a random link scoring table.

On the *x*-axis, it is the FPR (i.e., 1—specificity); on the *y*-axis, it is the TPR (i.e., sensitivity). Curves above the diagonal line are usually interpreted as good classifiers. The TPR of the link anomaly algorithm rises faster than its FPR while the random solution tends to have the same linear slope. The first half (0–0.5) is usually the most critical for performance, i.e., the quicker the curve rises, the better is the performance. It can be observed from Fig. 4 that both link anomaly IP and IP-port graphs have a steep curve at the beginning and IP graphs can achieve 0.8 TPR while only having less than 0.3 FPR. While both graphs are significantly better than the random case, the result suggests that IP graphs perform better than IP-port graphs even though IP graphs are simpler. However, this does not necessarily mean IP graphs always perform better than IP-port graphs as it may depend on the number of different anomalous types and the specific amount of port number information in the log data.

In reality, there is always a *tradeoff* among TP, TN, FP and FN. Parameters may be adjusted so that one measure increases while the other decreases, depending on how much one values each measure, e.g., is TP more valuable than TN, or is FP more costly than FN? While keeping a balance of these measures, we note that detecting some real anomaly is better than no detection at all. For example, if there are ten real attacks among a million connections, even detecting two of them can sometimes be satisfying because those stealth attacks could go by undetected without the proper tools. In a busy network with millions of connections each day, suggesting only a few suspicious links to the network administrator for further investigation can be very helpful, with understanding that a small portion of those suggestions may be false, which can be quickly examined and removed by the administrator.

To illustrate how to choose a good threshold of link anomaly scores, Figs. 5 and 6 compare the performance impact in terms of TP/FP/TN/FN. Specifically, Fig. 5 illustrates how various threshold percentages (from left to right) affect the total number of FP and TN (top chart) and the total number of TP and FN (bottom

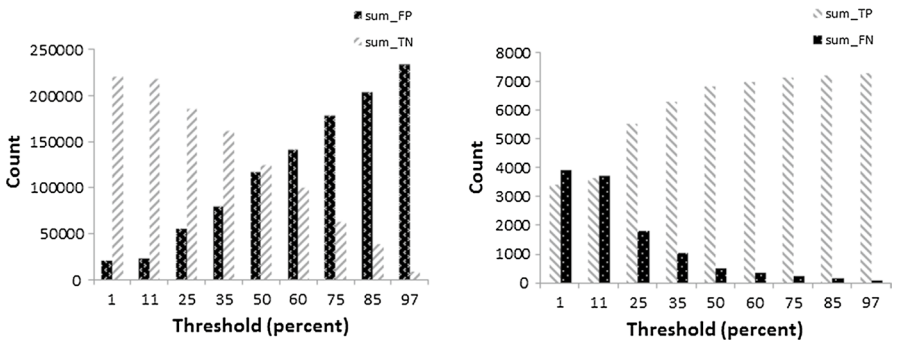


Fig. 5 Comparison of various thresholds. 35% may be a choice for a good balance of TP, FP, TN and FN. **a** FP versus TN, **b** TP versus FN

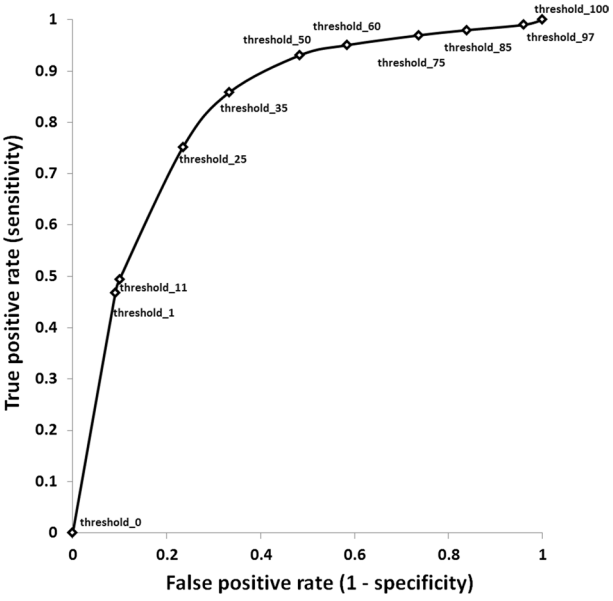


Fig. 6 An alternative comparison of various thresholds in terms of ROC curve, which is consistent with Fig. 5, i.e., 35% threshold may achieve a high TPR while keeping a reasonable FPR

chart). If the threshold is extremely low, there will be many TN and few FP, which is a good thing, but there will also be higher FN and lower TP, which is not good. On the other side, if the threshold is extremely high, there will be many FP and few TN, which is not good, but there will also be many TP and few FN, which is good. So the choice is to strike for a good balance of the tradeoff to achieve as high TP and TN as possible while keeping a reasonable amount of FP and FN, e.g., 35% threshold might be a candidate in this example. As an alternative view, Fig. 6 shows the ROC curve of TPR/FPR ratio with data points resulting from varying thresholds. Consistent with Fig. 5, 35% seems to be a good point to choose with about 0.85 TPR and less than 0.35 FPR.

We measure the accuracy, error rate, sensitivity and specificity of our link anomaly detection algorithms in Fig. 7a–d. We perform 300 rounds of link anomaly detections with the sliding window starting from the beginning of the dataset and moving towards the end. For each time window, a prior graph of all network traffic in the previous time window is built for learning and a current graph for all traffic in the current time window is built for testing. The IDS log is used for verification purpose. For each round of testing, we record the accuracy, error rate and sensitivity for the detection system. From Fig. 7a, it is clear that the accuracies, or the percentages of connections that are correctly identified as either normal or abnormal, are consistently centered around 0.76. There has been no case where the accuracy is less than 0.72. The chart suggests that the proposed link anomaly algorithm is effective in accurate identification of the nature of all network connections (normal or abnormal). In contrast, Fig. 7b shows that the error rates, or the percentages of incorrectly

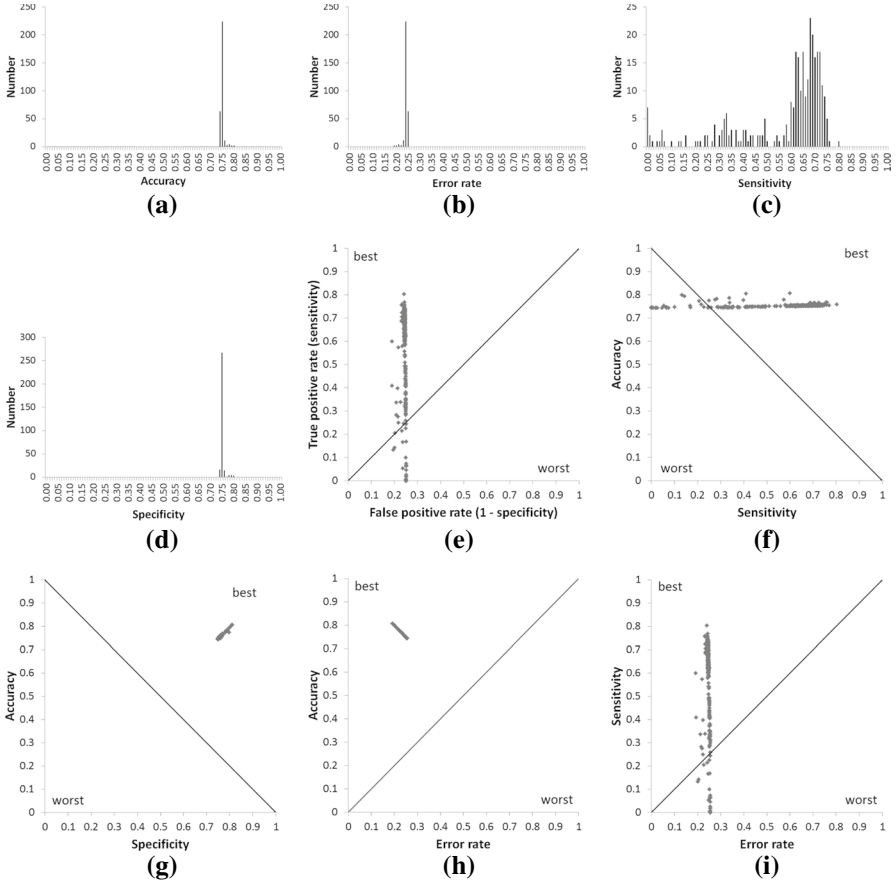


Fig. 7 Comparative study of dynamic link anomaly detection using combinations of metrics. **a** Accuracies, **b** error rates, **c** sensitivities, **d** specificities, **e** TPR versus FPR, **f** accuracy versus sensitivity, **g** accuracy versus specificity, **h** accuracy versus error rate, **i** sensitivity versus error rate

identified connections (normal or abnormal), of all link anomaly detections are consistently centered around 0.24, with no case having more than 0.25 error rate.

Sensitivities, or the true positive rates (TPRs), show a lot more variation, as shown in Fig. 7c. While the distribution has a long head, the majority of the sensitivities of the 300 link anomaly detection cases are between 0.6 and 0.75. Sometimes being able to detect true anomalies, even though just a few, can be valuable. For instance, suppose there are 10 truly problematic connections hiding in vast number of log files. Without any tool, an administrator may find nothing suspicious. With the assistance of the link anomaly detection algorithm, even finding just one true problematic link can be critical in fault localization or intrusion detection. The results suggest that approximately 70% of those real problematic links can be found by the link anomaly detection algorithms. Finally, Fig. 7d shows that the specificities, or true negative rates (TNRs), are uniformly distributed around 0.75.

We further study the tradeoff of the link anomaly detection algorithms through various combinations of the major performance metrics, as shown in Fig. 7e–i. For easy comparison, we mark the corners as "best" and "worst," indicating which side is ideal. Each data point in these 2D charts means the result from one of the 300 rounds of link anomaly detection tasks with sliding windows described earlier.

Figure 7e shows that the majority of link anomaly detections are in the best zone with higher TPR than FPR with only a few exceptions. Some TPRs are as high as 0.8. In all cases, no FPR is greater than 0.27. In the theoretically perfect case, TPR = 1 and FPR = 0, thus the best corner. Figure 7f compares accuracy versus sensitivity, or in other words, accuracy in identifying all connections versus accuracy in identifying only true abnormal connections. The accuracies for all connections are consistently above 0.75 while the majority of data points have satisfying sensitivity rates ranging from 0.3 to 0.8, mostly are over 0.5. Due to the visualization constraint, many dots overlap in the upper-right area.

Figure 7g shows the accuracy versus specificity (TNR). The results suggest the algorithms have consistent good performance, with all points near the upper-right (0.8, 0.8) best zone. The near perfect results are due to the fact that most normal (negative) connections are correctly identified. Figure 7h shows accuracy versus error rate. All cases are proved to be having high accuracy and low error rates near the (0.75, 0.25) location, close to the best zone. The results clearly indicate the link anomaly prediction system performs consistently well in accurately identifying all network connections while keeping a low error rate. Finally, Fig. 7i shows results in sensitivity versus error rate. No error rates are more than 0.25. The dominating majority of testing points suggest much higher sensitivities (TPRs) than the error rates, thus in the better zone.

5 Case Study

In this section, we conduct a case study over VAST Challenge'11 (MC2) dataset to show how such system may be used to analyze link anomalies as well as to identify interesting events in the network log data for situational awareness. Using a typical system administrator assisted by the link anomaly detection algorithms in the case study, we demonstrate the work flow for log examination and security investigation to show how the link anomaly algorithms may help to find out anomalies and reach the final conclusion.

Suppose the company's network and system administrator, Adam, needs to examine his network traffic flow data for potential bad activities. Since he has no clue what specific badness he is looking for, he is unable to search for any specific pattern in the dataset. Therefore, he loads the log files into the link anomaly detection system, which extracts from the raw data a few attributes such as timestamps, operation types, source and destination IP addresses, source and destination ports, etc. The system will automatically construct two types of connectivity graphs with either IP or IP-port nodes. The two graphs give different insights on the networks with a tradeoff of granularity and complexity.

Adam first defines parameters for the sliding windows to analyze the data since graphs are dynamically constructed based on connections during each time period. The sliding window size is set as 15 min, which means the prior graph includes all connections during the past 15-min window. The time zone size or sliding interval is 5 min meaning each time the system will move the window forward for 5-min amount of traffic, and the current graph is constructed based on the most recent traffic during the past 5 min. Having a too wide time window risks overfitting the data in the prior graph and possibly including data noise, e.g., the attack traffic. The normalized connectivity scores in the ranking table will be close to one suggesting that almost all connections in the current graph are normal. Having a too narrow window risks not having enough normal traffic to learn, and the scores will be close to zero suggesting that most connections are suspicious. Fifteen minutes seem to be a good balance to generate appropriate distribution of connectivity scores in the rank tables. Figures 8 and 9 show a comparison between the distributions of time frequency scores generated with a larger window size and a smaller window size. Figure 8 shows the time frequency score distribution of an IP-port graph starting at 2011-04-13 12:07:00 in a 1-min window size in the VAST'11 challenge data. The distribution is more uniform with extremely low values possibly due to lack of records, thus is hard to be used to differentiate connections for link anomaly detection. Figure 9 increases the graph size by enlarging the time window size to 15 min. It is obvious that larger variants make it easier to differentiate connections and are more ideal for link anomaly detection.

In order to identify suspicious connections, Adam looks at the distribution of scores (Fig. 10) in the connectivity ranking table and puts connections into anomalous set. For instance, Fig. 10 partially shows a snapshot of a ranking table on day 1 (2011-4-13). The connections on the left side have very low scores meaning they are

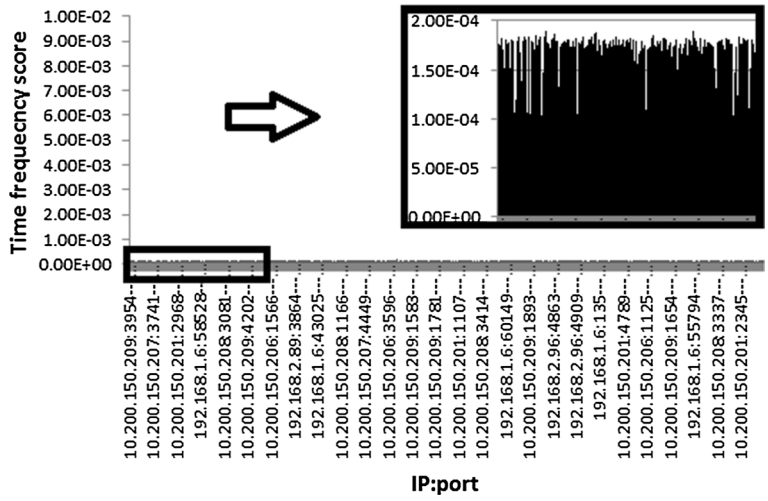


Fig. 8 The time frequency score distribution of an IP-port graph in a 1-min window size. The right box area is a magnification of the left box area. Links' values are mixed at the same extremely low level, which is hard to distinguish or predict for link anomalies

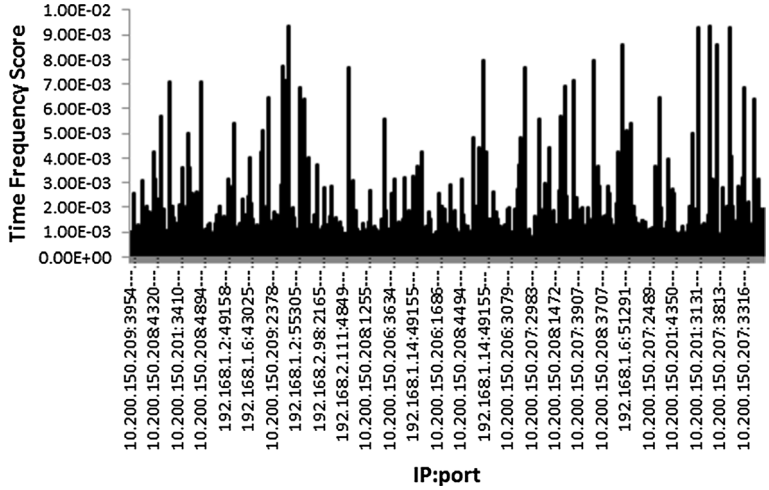
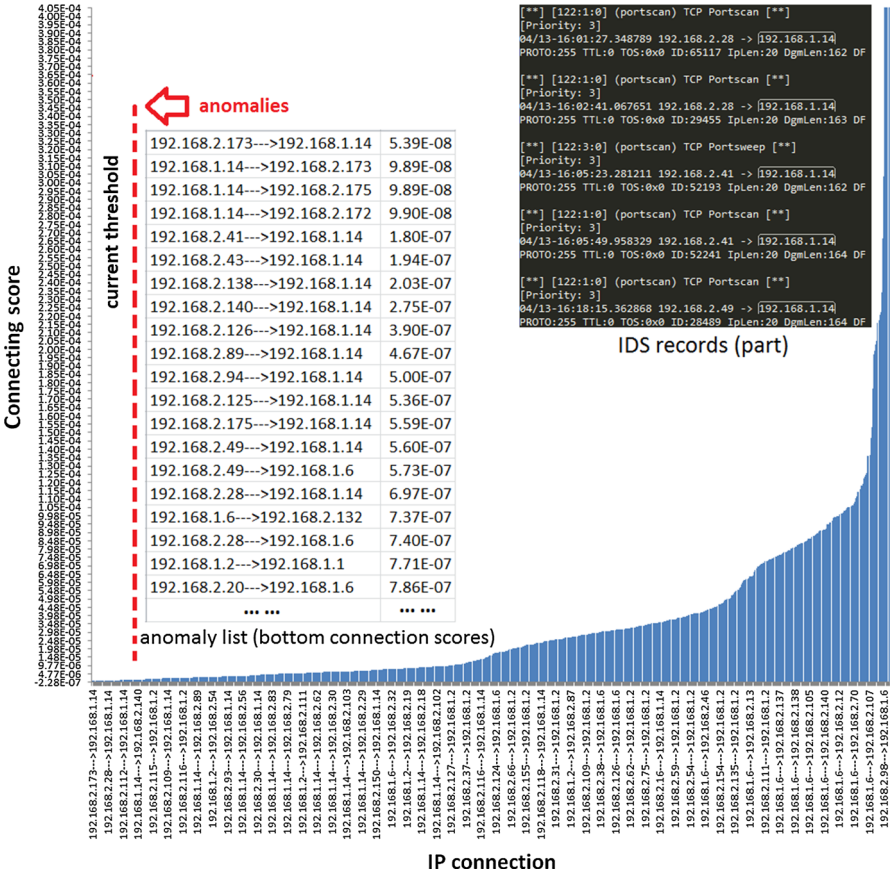


Fig. 9 The time frequency distribution of the same graph in Fig. 8 with a 15-min window size. Larger variants make it easier to differentiate connections for link anomaly detection

highly unlikely to appear while the connections on the right have very high scores meaning they are very likely to appear. Due to the space limitation on the x-axis, only a few IP addresses are actually shown. Adam simply draws the top 10% as the threshold line and marks connections to the *left* of the threshold as anomalies, and puts them into the anomaly list. The anomaly detection is based on the fact that those connections that have the least connection probabilities actually appeared. A group of machines with IP addresses ranging from 192.168.2.11 to 192.168.2.138 connect to three particular machines with IP 192.168.1.2, 192.168.1.6, and 192.168.1.14, which are underlined in red. This is also shown in the list included in Fig. 11. In addition, workstations 192.168.2.171-175 are also the sources for many port scans to other hosts in the subnet. These link anomalies are confirmed as compromised machines starting to conduct port scanning and DDoS attacks in the IDS log.

Turning to the other side of the connection score distribution, Adam examines Fig. 12. The connections to the right of the threshold have very high connection scores meaning they are very likely to appear. However, from the firewall log, Adams finds that these connections are actually torn down, e.g., from 04/13/2011 12:37 to 04/13/2011 12:52, and therefore disappear from the network graph in the following snapshot. This is certainly suspicious. Adam marks these connections as anomalies and puts them into the anomaly list, as shown in the figure. The sources are from 10.200.150.201, 206-9. From the IDS log, Adam confirms that there is actually an attempted denial of service (DoS) attack against the corporate web server 172.20.1.15, e.g., at 04/13/2011 11:43:39, for links which source IP is 10.200.150.209. As a result of numerous DoS attacks beginning at 11:39 that day, external systems try to disrupt communications with the corporate web server 172.20.1.5 and finally make it break down for a short time. While these examples in the case study are results of malicious attacks, the dynamic link anomaly analytic system could potentially detect any abnormal connections that are not due to



192.168.2.173--->192.168.1.14	5.39E-08
192.168.1.14--->192.168.2.173	9.89E-08
192.168.1.14--->192.168.2.175	9.89E-08
192.168.1.14--->192.168.2.172	9.90E-08
192.168.2.41--->192.168.1.14	1.80E-07
192.168.2.43--->192.168.1.14	1.94E-07
192.168.2.138--->192.168.1.14	2.03E-07
192.168.2.140--->192.168.1.14	2.75E-07
192.168.2.126--->192.168.1.14	3.90E-07
192.168.2.89--->192.168.1.14	4.67E-07
192.168.2.94--->192.168.1.14	5.00E-07
192.168.2.125--->192.168.1.14	5.36E-07
192.168.2.175--->192.168.1.14	5.59E-07
192.168.2.49--->192.168.1.14	5.60E-07
192.168.2.49--->192.168.1.6	5.73E-07
192.168.2.28--->192.168.1.14	6.97E-07
192.168.1.6--->192.168.2.132	7.37E-07
192.168.2.28--->192.168.1.6	7.40E-07
192.168.1.2--->192.168.1.1	7.71E-07
192.168.2.20--->192.168.1.6	7.86E-07
... ...	... ...

Fig. 10 The score distribution for all connections appeared in the current graph. The connections to the left of the threshold are anomalous since they should not appear but actually appear. It turns out hosts such as 192.168.1.14 are attacked by port scans as verified by the IDS records

connection	connection score	connection	connection score	connection	connection score
192.168.2.173--->192.168.1.14	5.39E-08	192.168.2.56--->192.168.1.14	9.16E-07	192.168.1.2--->192.168.2.140	1.42E-06
192.168.1.14--->192.168.2.173	9.89E-08	192.168.2.124--->192.168.1.14	9.40E-07	192.168.2.59--->192.168.1.14	1.42E-06
192.168.1.14--->192.168.2.175	9.89E-08	192.168.1.6--->192.168.2.155	9.42E-07	192.168.2.158--->192.168.1.14	1.47E-06
192.168.1.14--->192.168.2.172	9.90E-08	192.168.2.40--->192.168.1.14	9.68E-07	192.168.1.6--->192.168.2.130	1.48E-06
192.168.2.41--->192.168.1.14	1.80E-07	192.168.1.2--->192.168.2.173	9.72E-07	192.168.2.15--->192.168.1.14	1.49E-06
192.168.2.43--->192.168.1.14	1.94E-07	192.168.1.2--->192.168.2.175	9.97E-07	192.168.1.6--->192.168.2.24	1.49E-06
192.168.2.138--->192.168.1.14	2.03E-07	192.168.2.114--->192.168.1.14	1.02E-06	192.168.2.139--->192.168.1.14	1.50E-06
192.168.2.140--->192.168.1.14	2.75E-07	192.168.2.112--->192.168.1.14	1.07E-06	192.168.1.6--->192.168.2.39	1.55E-06
192.168.2.126--->192.168.1.14	3.90E-07	192.168.1.14--->192.168.2.45	1.07E-06	192.168.2.103--->192.168.1.14	1.62E-06
192.168.2.89--->192.168.1.14	4.67E-07	192.168.1.2--->192.168.2.45	1.07E-06	192.168.2.51--->192.168.1.14	1.67E-06
192.168.2.94--->192.168.1.14	5.00E-07	192.168.2.145--->192.168.1.2	1.09E-06	192.168.2.100--->192.168.1.14	1.68E-06
192.168.2.125--->192.168.1.14	5.36E-07	192.168.1.14--->192.168.2.138	1.13E-06	192.168.2.74--->192.168.1.2	1.68E-06
192.168.2.175--->192.168.1.14	5.59E-07	192.168.1.6--->192.168.2.135	1.14E-06	192.168.2.32--->192.168.1.6	1.77E-06
192.168.2.49--->192.168.1.14	5.60E-07	192.168.1.2--->192.168.2.138	1.16E-06	192.168.2.66--->192.168.1.6	1.77E-06
192.168.2.49--->192.168.1.6	5.73E-07	192.168.1.2--->192.168.2.43	1.18E-06	192.168.2.115--->192.168.1.2	1.78E-06
192.168.2.28--->192.168.1.14	6.97E-07	192.168.2.133--->192.168.1.14	1.19E-06	192.168.2.75--->192.168.1.14	1.80E-06
192.168.1.6--->192.168.2.132	7.37E-07	192.168.1.2--->192.168.2.172	1.20E-06	192.168.2.62--->192.168.1.14	1.80E-06
192.168.2.28--->192.168.1.6	7.40E-07	192.168.1.14--->192.168.2.43	1.22E-06	192.168.2.142--->192.168.1.2	1.83E-06
192.168.1.2--->192.168.1.1	7.71E-07	192.168.1.6--->192.168.2.44	1.25E-06	192.168.1.2--->192.168.2.126	1.87E-06
192.168.2.20--->192.168.1.6	7.86E-07	192.168.1.6--->192.168.2.136	1.30E-06	.....	.....
192.168.2.128--->192.168.1.14	8.13E-07	192.168.2.54--->192.168.1.14	1.30E-06		
192.168.1.14--->192.168.2.94	8.36E-07	192.168.2.61--->192.168.1.14	1.36E-06		
192.168.2.123--->192.168.1.14	8.60E-07	192.168.1.14--->192.168.2.140	1.37E-06		

Fig. 11 Among the src/dst node pairs with least possibilities, the actual appeared connections are highlighted to indicate confirmed port scan activities from the IDS

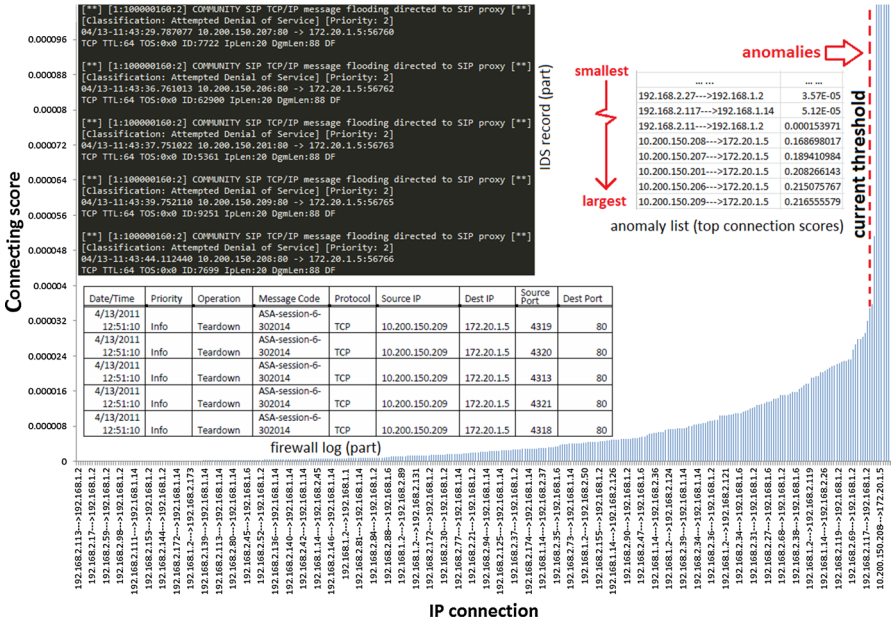


Fig. 12 The connections to the right of the threshold are anomalous since they have high connection scores but are actually torn down and disappear from the current network graph. It turns out to be an attempted DoS attack from machines 10.200.150.201, 206-9 against the corporate web server 172.20.1.5

malicious activities, e.g., due to hardware failures or misconfigurations. Therefore, it may also benefit other network management tasks such as troubleshooting and diagnosing.

6 Conclusion

In this paper, we study an important yet challenging research problem in dynamic network analysis (DNA), i.e., dynamic link anomaly analysis (DLAA). One major challenge of dynamic networks is making educated guess of suspicious network changes. We introduce the network similarity metrics and sliding time windows for data stream mining in order to incorporate the link anomaly detection into the dynamic network analysis. To make our algorithms generic, we utilize spatial–temporal information, i.e., the topological similarity measures and weighted time frequency functions. We formally define the complete situations for DLAA and verification. Through tradeoff analysis and case study, we demonstrate the proposed dynamic link anomaly detection framework provides the capability to construct effective knowledge structures by measuring the security levels of dynamic networks, and filtering anomalous network links. We believe the DLAA algorithm is useful in network security management and has potential impact on the analysis of many other types of networks as well. Our future work is to conduct a comparative study on how the generic algorithms perform in face of dynamics of various

networks in addition to enterprise networks, e.g., cloud networks, sensor networks, mobile networks, social networks, etc.

Acknowledgements This work was supported in part by CMU Early Career Grant (C61920) and ASEE Fellowship U.S. Air Force SFFP Program. We thank Dr. Keesook J. Han who provided insight and expertise that greatly assisted the research.

References

1. Carley, K., Pfeffer, J.: Dynamic network analysis (DNA) and ORA. In: Proceedings of the 2nd International Conference on Cross-Cultural Decision Making: Focus 2012 , San Francisco, CA, July 21–25 (2012)
2. García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E.: Anomaly-based network intrusion detection: techniques, systems and challenges. Comput. Secur. **28**(1–2), 18–28 (2009)
3. Ahmed, M., Mahmood, A.N., Hu, J.: A survey of network anomaly detection techniques. J. Netw. Comput. Appl. **60**, 19–31 (2016)
4. Lü, L., Zhou, T.: Link prediction in complex networks: a survey. Phys. A Stat. Mech. Appl. **390**(6), 1150–1170 (2011)
5. Liben-Nowell, D., Kleinberg, J.: The link prediction problem for social networks. In: The 12th International Conference on Information and Knowledge Management (CIKM), New Orleans, LA, November 3–8 (2003)
6. Hasan, M.A., Chaoji, V., Salem, S., Zaki, M.: Link prediction using supervised learning. In: SIAM Workshop on Link Analysis, Counterterrorism and Security with SIAM Data Mining Conference, Bethesda, MD (2006)
7. Lakhina, A., Crovella, M., Diot, C.: Characterization of network-wide anomalies in traffic flows. In: Proceedings of the 4th ACM SIGCOMM Conference on Internet measurement, Ser. IMC'04, pp. 201–206. ACM, New York (2004)
8. Szmit, M., Szmit, A., Adamus, S., Bugala, S.: Usage of Holt–Winters model and multilayer perceptron in network traffic modelling and anomaly detection. Informatica **36**(4), 359–368 (2012)
9. Kwon, D., Kim, H., Kim, J., Suh, S.C., Kim, I., Kim, K.J.: A survey of deep learning-based network anomaly detection. Clust. Comput. **4**, 1–13 (2017)
10. Sommer, R., Paxson, V.: Outside the closed world: on using machine learning for network intrusion detection. In: IEEE Symposium on Security and Privacy, Oakland, CA, vol. 16, no. 19, pp. 305–316 (2010)
11. Sun, J., Qu, H., Chakrabarti, D., Faloutsos, C.: Neighborhood formation and anomaly detection in bipartite graphs. In: IEEE International Conference on Data Mining (ICDM '05), Houston, TX (2005)
12. Akoglu, L., McGlohon, M., Faloutsos, C.: OddBall: spotting anomalies in weighted graphs. In: The 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Hyderabad, India (2010)
13. Huang, Z., Zeng, D.: A link prediction approach to anomalous email detection. In: IEEE International Conference on Systems, Man, and Cybernetics, Taipei, Taiwan, October 8–11 (2006)
14. Liu, L., Zuo, W.L., Peng, T.: Detecting outlier pairs in complex network based on link structure and semantic relationship. Expert Syst. Appl. **69**, 40–49 (2017)
15. Carley, K.M.: ORA: a toolkit for dynamic network analysis and visualization. In: Alhajj, R., Rokne, J. (eds.) Encyclopedia of Social Network Analysis and Mining, pp. 1219–1228. Springer, New York (2014)
16. Parraguez, P., Eppinger, S.D., Maier, A.M.: Information flow through stages of complex engineering design projects: a dynamic network analysis approach. IEEE Trans. Eng. Manag. **62**(4), 604–617 (2015)
17. Javed, M.A., Younis, M.S., Latif, S., Qadir, J., Baig, A.: Community detection in networks: a multidisciplinary review. J. Netw. Comput. Appl. **108**, 87–111 (2018)
18. Yasami, Y., Safaei, F.: A statistical infinite feature cascade-based approach to anomaly detection for dynamic social networks. Comput. Commun. **100**(C), 52–64 (2017)
19. Beck, F., Burch, M., Diehl, S., Weiskopf, D.: A taxonomy and survey of dynamic graph visualization. Comput. Graph. Forum **36**(1), 133–159 (2017)

20. Katz, L.: A new status index derived from sociometric analysis. Psychometrika **18**(1), 39–43 (1953)
21. Getoor, L., Diehl, C.P.: Link mining: a survey. ACM SIGKDD Explor. Newsl. **7**(2), 3–12 (2005)
22. Chakrabarti, D., Faloutsos, C.: Graph mining: laws, generators, and algorithms. ACM Comput. Surv. **38**(2), 1–69 (2006)
23. Lichtenwalter, R.N., Lussier, J.T., Chawla, N.V.: New perspectives and methods in link prediction. In: The 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington DC, pp. 243–252 (2010)
24. O'Madadhain, J., Hutchins, J., Smyth, P.: Prediction and ranking algorithms for event-based network data. ACM SIGKDD Explor. Newsl. **7**(2), 23–30 (2005)
25. Almansoori, W., Gao, S., Jarada, T.N., Elsheikh, A.M., Murshed, A.N., Jida, J., Alhajj, R., Rokne, J.: Link prediction and classification in social networks and its application in healthcare and systems biology. Netw. Model. Anal. Health Inform. Bioinform. **1**(1–2), 27–36 (2012)
26. Potgieter, A., April, K., Cooke, R., Osunmakinde, I.: Temporality in link prediction: understanding social complexity. Sprouts: working papers on information systems, vol. 7, no. 9 (2007)
27. Rattigan, M.J., Jensen, D.: The case for anomalous link discovery. ACM SIGKDD Explor. Newsl. **7**(2), 41–47 (2005)
28. Wan, X., Milios, E., Kalyaniwalla, N., Janssen, J.: Link-based anomaly detection in communication networks. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT '08), pp. 402–405 (2008)
29. Takahashi, T., Tomioka, R., Yamanishi, K.: Discovering emerging topics in social streams via link-anomaly detection. IEEE Trans. Knowl. Data Eng. **26**(1), 120–130 (2014)
30. Camacho, J., Padilla, P., García-Teodoro, P., Díaz-Verdejo, J.: A generalizable dynamic flow pairing method for traffic classification. Comput. Netw. **57**(14), 2718–2732 (2013)
31. Fire, M., Tenenboim, L., Lesser, O., Puzis, R., Rokach, L., Elovici, Y.: Link prediction in social networks using computationally efficient topological features. In: SocialCom/PASSAT, pp. 73–80. IEEE (2011)
32. Liao, Q., Striegel, A.: Intelligent network management using graph differential anomaly visualization. In: Network Operations and Management Symposium (NOMS), pp. 1008–1014. IEEE (2012)
33. Foster, K.C., Muth, S.Q., Potterat, J.J., Rothenberg, R.B.: A faster Katz status score algorithm. Comput. Math. Organ. Theory **7**(4), 275–285 (2001)

Tao Zhang is a Ph.D. student in the Computer Science Department at College of William and Mary. He received his M.S. degree in Computer Science from Central Michigan University. His research interests include computer security, specifically in architectural support for security, anomaly detection, and cybersecurity data analysis and visualization.

Qi Liao is an Associate Professor of Computer Science at Central Michigan University. He received his M.S. and Ph.D. in Computer Science and Engineering from the University of Notre Dame. He graduated with a B.S. and departmental distinction in Computer Science from Hartwick College with a minor concentration in Mathematics. Dr. Liao's research interests include computer security, machine learning, visual analytics, and economics/game theory at the intersection of network usage and cybersecurity.