A Model-based Approach to Anomaly Detection in Software Architectures

Hemank Lamba, Thomas J. Glazier,
Bradley Schmerl, Javier Cámara, David Garlan, Jürgen Pfeffer
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA, USA
{hlamba,tglazier,schmerl,jcmoreno,garlan,jpfeffer}@cs.cmu.edu

ABSTRACT

In an organization, the interactions users have with software leave patterns or traces of the parts of the systems accessed. These interactions can be associated with the underlying software architecture. The first step in detecting problems like insider threat is to detect those traces that are anomalous. Here, we propose a method to find anomalous users leveraging these interaction traces, categorized by user roles. We propose a model based approach to cluster user sequences and find outliers. We show that the approach works on a simulation of a large scale system based on and Amazon Web application style.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

Keywords

anomaly detection, model-based graph clustering

1. INTRODUCTION

In the contemporary corporate setting, having anomalous users interacting with software is potentially disastrous and can cost organizations millions of dollars. For instance, Cummings et al. [1] studied 80 cases of fraud in the financial services sector. They found that in most cases, the users interacted with systems they had access to, due to their roles in the companies – however, they were behaving anomalously, compared to how they had behaved before, or how others in similar roles behaved. They discovered that the impact of such cases was as high as 28 million dollars. Similar results were found in other sectors as well - government [3], the information technology and telecom sector [4] and critical infrastructure sectors [2]. Anomalous behavior can be caused by the existence of compromised user accounts, rogue users, or by less knowledgeable users who break things at random.

Automated approaches are needed to analyze user actions and determine which of them are anomalous. To be applicable, anomaly detection algorithms need to work for modern and common systems. They need to (a) scale to the size of systems and number of users, (b) accurately detect anomalies in the presence of diverse and nuanced user roles, (c) account for the complex structure and organization of today's systems, (d) take into consideration that users have complex interactions with software systems that involve their interacting with multiple parts of the system, and (e) be understandable at a level that enables system administrators to determine which anomalies are likely to represent malicious or dangerous behavior.

In this poster, we describe an approach that improves upon existing anomaly detection approaches, since: (a) it can automatically infer or provide intuition about organizational roles based on traces of user behavior through complex software systems, (b) it can represent complex systems at a software architecture level of abstraction, enabling scalability and interpretability, and (c) it can be used in the context of modern, distributed software systems of large scale with many users.

The key ideas behind our approach are:

- Use the architectural description of software systems as the primary graph on which to detect anomalies. This captures the idea that software systems are not monolithic entities, and that some anomalies can only be detected by understanding the ways in which users interact with many parts of the system;
- A novel model-based graph clustering algorithm that takes into consideration the sequence of interactions with components in the system to determine clusters based on learned roles. This approach detects anomalies by the distance that a particular trace is from any of the clusters.

2. MODEL-BASED ANOMALY DETECTION

For the task of finding anomalous patterns from sequences (which are represented as paths over the software architectural graph), we propose a clustering-based framework. The clustering-based approach first clusters all the sequences into a number of clusters, and then those sequences that are weakly linked to their respective cluster are marked as anomalous or outliers.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

HotSoS '16 April 19-21, 2016, Pittsburgh, PA, USA

© 2016 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-4277-3/16/04.

DOI: http://dx.doi.org/10.1145/2898375.2898401

Each user has a role to play in the organization, and we assume that the activities a user carries out during the session is heavily dependent on her role. Once the role is inferred, a user can choose a sequence of activities from the sequence distribution conditional to that role. This could be written as a generative model, in the following way:

1. A user implicitly chooses what cluster she belongs to. This assignment may depend on her role, or the task that she wants to perform.

2. Given the cluster, then her behavior, in the form of a sequence of activities, is generated from the model from some activity distribution related to that cluster.

Each cluster can be represented by a transition matrix that captures the probability of moving from one state to another by a user. These transition matrices are represented by $\theta = (\theta_1, \theta_2, \theta_3 ....\theta_k)$. Algorithm 1 starts by initializing all the transition matrices randomly, and also initializing sequences into clusters randomly. Then, it keeps on alternating with E-step and M-Step until it reaches convergence. E-Step assumes that transition matrices are fixed, and assigns each sequence to a cluster that maximizes the probability of observing that sequence in the given data. M-Step assumes that cluster assignments are fixed and updates the transition matrix that can best explain the cluster assignments. For experimental purposes, we have set the convergence condition as one in which there is little change (10^{-4}) in the log-likelihood of the data.

Algorithm 1 Model-based sequence clustering algorithm

Input: Number of clusters(K) and sequence data
Output: Cluster assignments and scores

```
Initialize cluster assignments z_1, z_2, ...z_n
Initialize transition probability matrices θ_1, θ_2, ...θ_k
while Not Converged do
    E-Step: Assign each sequence to most likely cluster
    for i = 0 to n do
        z_i = arg max_{z_i} p(x|c_i)
    end for
    M-Step:Update transition matrices θ_1, θ_2, ...θ_k
    for z = 1 to k do
        for all sequences in cluster z do
            Count number of transition edges from a to b in
            the sequence.
            Update transition matrix θ_z
        end for
    end for
end while
```

However, the model also requires as input the number of clusters. For large datasets, figuring out the number of clusters can be a problem because sequence clustering is sensitive to this parameter. Using a different number of clusters as input could potentially produce very different results, which might not be consistent with what we are trying to capture, or explain the data well. To come up with an appropriate number of clusters, we employ the bayesian information criterion [5].

To find outliers, consider the probability value with which each sequence may be part of the each cluster. The sequences with high membership scores, which form clusters of their own, are candidates for anomalous sequences.

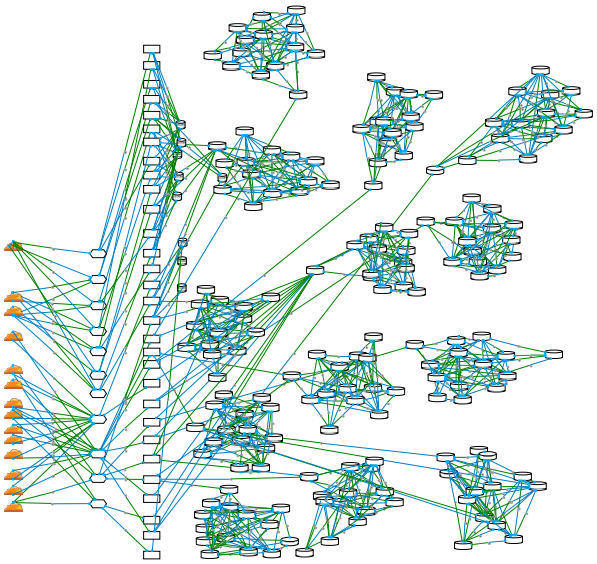


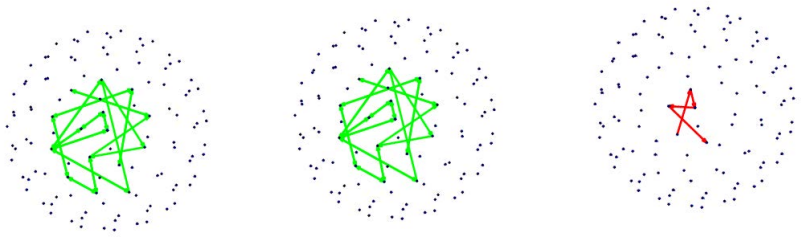
Figure 1: Example architecture of Amazon Web application.

3. RESULTS

We apply our sequence-based clustering model to the simulated dataset of a representative system developed using Amazon Web Services (shown in Figure 1), with an injected anomaly. To find an appropriate number of clusters for our dataset, we run a scan from 2 to 100 using BIC. We found that the appropriate number of clusters was 24. We cluster user sequences into those 24 clusters, obtaining the scores for each of the given sequences. The distribution of these scores is shown in Fig 3. The figure clearly shows that the outlier is to the extreme right of the distribution. This point is the outlier score of the injected anomalous sequence. In Fig 2, we present the sequence of activities for some clusters, including the anomalous one. In each of these sub-figures, we represent the graph, including an overlay that represents the sequence of activities as a path on the network. Each node represents a unit in the software architecture, and each edge represents the transition between the consecutive nodes in the sequence. As we mentioned earlier, the organizational roles determine the sequence of activities that a user performs. In our algorithm, the latent clusters formed capture these organizational roles. For instance, cluster12 in our simulation captures a role in which most of the requests are being transferred from both European and US CDN's to the same set of distributed databases. However, the injected anomaly, which forms a cluster of its own, behaves very differently because the trace starts on an interaction with the application server.

4. DISCUSSION AND FUTURE WORK

In this poster we have shown the use of a new, model-based clustering algorithm that considers sequences of interaction with components in a software architecture and user roles when clustering. The approach addresses the challenges of applying existing feature- or community-based graph clustering algorithms by considering user interactions with each component in the software architecture, and constructing paths, or traces, that users create when interacting with software. We have shown that our approach can find anomalous user traces in a realistic example of a cloud-based



(a) Cluster 12 (b) Cluster 22 (c) Cluster 0

Figure 2: User paths overlaid on the underlying architecture diagram. The first two figures show the combined sequences from all instances that form cluster 12, and 22, respectively. The fourth figure is the anomalous cluster 0.

software system. This is the first step in being able to identify insider threats.

While we have demonstrated that the model-based graph clustering algorithm performs well in a realistic setting, there are a number of limitations to our validation that we would like to strengthen in future work. In particular:

- The example system that we used, while based on realistic and reported scenarios, does not use actual real world data. Gaining access to the architectures of real systems, and to actual anomalous data, is extremely difficult.
- Because we are using simulated data, it is easy for us to get user traces. In real systems, this is likely to be more challenging.
- We have made the argument that this approach can find anomalies that other graph clustering approaches cannot, because they do not consider the structure of the system or the role of users in an organization. In future work, we would like to address this by implementing several of these alternative approaches on our data to more strongly verify this claim, and to be able to provide performance and complexity based comparisons between our approach and others.
- While we have shown we can detect an injected anomaly, we have not evaluated the characteristics of the algorithm in terms of its complexity, performance, or precision/recall.
- We have focused on a particular type of architectural style – a large web scale system – and a particular type of anomaly – staging data for exfiltration. While both of these have been constructed to be realistic as possible, future work would include evaluation of different types of applications and different types of security anomalies.

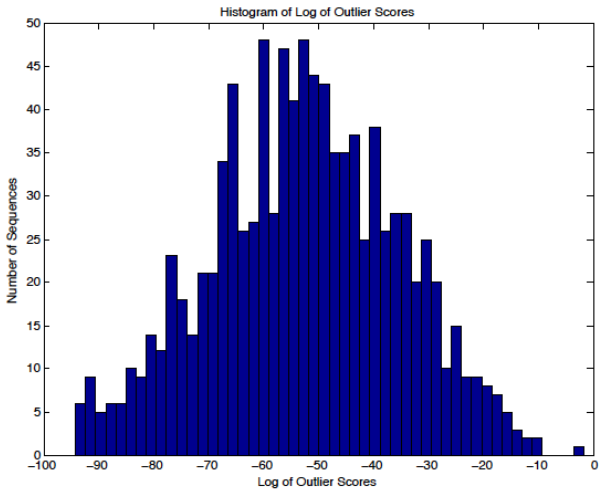


Figure 3: Histogram of the natural logarithm of the Outlier Scores for every sequence generated by the model based sequence clustering algorithm

Acknowledgments

This work is supported in part by the National Security Agency. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Security Agency or the U.S. government.

5. REFERENCES

[1] A. Cummings, T. Lewellen, D. McIntire, A. Moore, and R. Trzeciak. Insider threat study:illicit cyber activity involving fraud in the US financial services sector. *Special Report, CERT, Software Engineering Institute*, 2012.

[2] M. Keeney, D. Capelli, E. Kowalski, A. Moore, T. Shimeall, and S. Rogers. Insider threat study: Computer sabotage in critical infrastructure sectors. In *CERT Program and Software Engineering Institute*, 2005.

[3] E. Kowalski, D. Cappelli, and A. Moore. Insider threat study: Illicit cyber activity in the government sector. *Software Engineering Institute*, 2008.

[4] E. Kowalski, D. Cappelli, and A. Moore. Insider threat study: Illicit cyber activity in the information technology and telecommunications sector. *Software Engineering Institute*, 2008.

[5] G. E. Schwartz. Estimating the dimnesson of a model. *Annals of Statistics*, 6:461–464, 1978.