Detecting Insider Threats in Software Systems using Graph Models of Behavioral Paths

Hemank Lamba
Carnegie Mellon University
hlamba@cs.cmu.edu

Thomas J. Glazier
Carnegie Mellon University
tglazier@cs.cmu.edu

Bradley Schmerl
Carnegie Mellon University
schmerl@cs.cmu.edu

Jürgen Pfeffer
Carnegie Mellon University
jpfeffer@cs.cmu.edu

David Garlan
Carnegie mellon University
garlan@cs.cmu.edu

ABSTRACT

Insider threats are a well-known problem, and previous studies have shown that it has a huge impact over a wide range of sectors like financial services, governments, critical infrastructure services and the telecommunications sector. Users, while interacting with any software system, leave a trace of what nodes they accessed and in what sequence. We propose to translate these sequences of observed activities into paths on the graph of the underlying software architectural model. We propose a clustering algorithm to find anomalies in the data, which can be combined with contextual information to confirm as an insider threat.

Categories and Subject Descriptors

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

Keywords

insider threats, software architecture, graph models, path clustering

1. INTRODUCTION

In the contemporary corporate setting, the problem of insider threats is both a serious and a very likely risk for all organizations. The impact of insider attacks is huge; for instance a study by Cummings et al.[2] analyzed 80 cases of fraud in the financial services sector and found that some cases had an impact as high as $28 million. Similar empirical evidence was found in other sectors, including government, information technology, telecom sector, and critical infrastructure. Insider threats are more difficult to detect than external threats since insiders already have access to confidential information, and hence network intrusion detection approaches do not work. Furthermore, it is hard to come up with an efficient machine learning or pattern-mining approach since the frequency of such threats is low (there is low signal to noise ratio), when compared to the number of normal activity logs. Another challenge is the availability of such data for researchers to work on. The above mentioned limitations make the task of dealing with insider threats both challenging and interesting to solve.

One salient aspect of many insider threats is that people who access the system start behaving in a way that is different from their previous interactions and that those behaviors are malicious. Thus, a key part of detecting insider threats is (1) determining anomalous behavior and (2) detecting which behaviors are malicious. We use a simulator to generate artificial data representing the three layered web-tier architecture, and then over the data so generated, we apply a clustering algorithm.

2. APPROACH

To detect anomalous behaviors we first need to to detect behaviors. To do this, we can look at the software architecture of a system, and abstract events occurring in a running version of the system to behaviors in the architecture of the system [3]. Consider a common architecture style used by many organizations for their IT infrastructure - a three

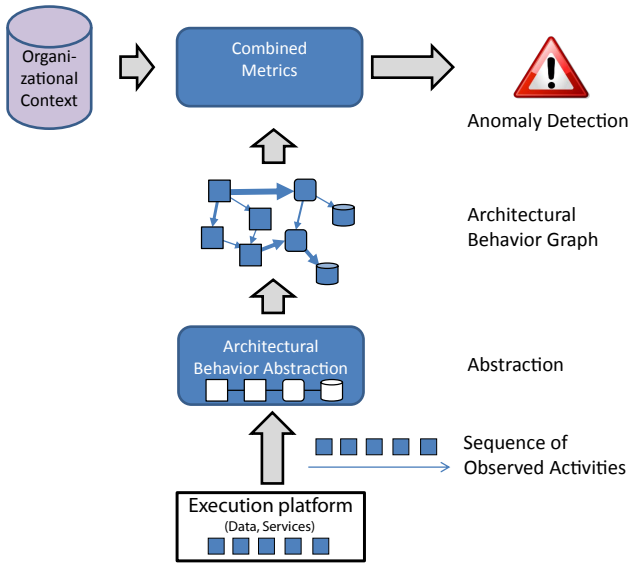


Figure 1: Overview of our approach.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Symposium and Bootcamp on the Science of Security '15, Urbana, Illinois USA
Copyright 2015 ACM ACM 978-1-4503-3376-4/15/04 ...$15.00.
http://dx.doi.org/10.1145/2746194.2746214

tiered web server where there are presentation components, business logic components and data access components. Every time a user interacts with such a system we can trace the events during the interaction as behaviors according to the architecture. All of these interactions are stored as records in the activity log. Every record includes which user or whose user-generated request accessed what resource and at what time. These traces can be represented by a triplet of *(userId, resourceId, timestamp)*. For every user, we can extract all his interactions from the activity log and call it user activity trace.

We can represent the entire activity log over the underlying software system as graphs (networks). A graph consist of a set of nodes and a set of edges connecting the nodes. The nodes are the resources, which users access (see Figure 1). Nodes are marked by labels indicating what type of component is accessed. Edges are generated by the order of user activities for instance, when a user accesses server 1 and then accesses server 2, we put an edge between server 1 and server 2. Thus a user's activity in the system can be represented as a path in the graph. This representation allows us to make use of various network properties to detect anomalous activities [4].

Figure 1 illustrates our approach. During runtime, a sequence of observed activities is translated into a path in the architectural behavior graph. Such a behavioral graph combines all users' paths (activities) into a single model, where patterns of behavior can be determined. The nodes of this graph are architectural design components and the directed edges store the probability with which a user move across these components. This graph is the result of many task executions, and is learned over time. When this graph of the software interactions is combined with organizational network data e.g., position, age, income, possible job terminations, performance reviews, etc.) we can answer questions such as whether the observed behavior is similar to a role-equivalent user, or can the observed anomalous behavior be explained by context change (new position, new project), or whether the observed anomalous behavior is malicious?

3. EVALUATION

To test our approach we use a simulation to generate behavior over a multi-tier architecture. The system is organized into tiers, each tier is organized into server groups, and each server group consists of multiple servers with identical functionality. Within each tier, each server group has an equal chance of being used and within each server group each server has an equal chance of being used. This simulates representative web based architectures with multiple types of front end / UI servers (e.g., WWW and FTP servers), multiple middle tier servers (e.g., different types of application servers), and multiple types of back end servers (e.g., databases and partner systems) with load balancers evenly distributing workload to servers in the group.

The users and their individual interactions are simulated as HTTP transactions, which collectively, for each user, constitute a single session. Each simulated HTTP transaction starts at the UI tier and then each server that the transaction interacts with determines if the transaction is returned to the previous tier eventually ending back to the user. Therefore, not each session will interact with each tier and those interactions are determined per server. This is representative of a wide variety of web-based transactions, as well as the tendency to cache items in memory on each server. Additionally, the time consumed by each interaction with each server is randomly determined, within parameters, to simulate variable processing time. The simulator produces a trace log with the unique user identifier, the unique server identifier, and the timestamp of the transaction. This information would usually be available in any web based system through consolidation of various web and application logs.

3.1 Clustering Behavioral Paths

Our clustering-based approach first clusters all the traces into an appropriate number of clusters. Next those sequences that are weakly linked to their respective cluster are marked as anomalous or outliers. We use an existing model-based approach to cluster sequences, and modify it to find anomalous sequences [1]. The model assumes that data is generated in the following way: first a user implicitly chooses what cluster he belongs to, which might be dependent on his role, or task. Given this cluster assignment, his behavior is generated from the behaviors that are highly probable in that particular cluster. We use the expected maximization (EM) approach to output latent cluster assignments that maximize the likelihood of observed data. Anomalies can be marked as sequences that are weakly connected to their respective cluster.

4. CONCLUSIONS AND FUTURE WORK

With our approach we present a step toward finding insider threats in a large organization by detailing a way that a clustering-based approach over behavioral graphs can be used to detect anomalous behaviors in a software system. Activity traces of users over software architecture can be represented as graphs, thus allowing us to use graph-based path clustering algorithms to find anomalies. This approach describes one aspect of detecting insider threats. Once we have detected anomalous user behaviors, we need to be able to determine whether those behaviors are threats. We believe that we can use additional organizational and software architectural information to help to determine this. Another aspect of future work is to investigate the ability to calculate and use these metrics in real time. We believe that the algorithms that we have described here can be made incremental, but verifying this is an area of future work. Finally, the work described here uses a representative simulation of a common three-tiered architecture. We plan to apply this approach to real data sets that are becoming available, to further validate the results.

5. REFERENCES

[1] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of navigation patterns on a web site using model-based clustering. In *Proceedings of ACM SIGKDD*, pages 280–284, 2000.

[2] 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.

[3] D. Garlan and B. Schmerl. Architecture-driven modelling and analysis. In *Proceedings of SCS'06*, 2006.

[4] M. Hennig, U. Brandes, J. Pfeffer, and I. Mergel. *Studying Social Networks. A Guide to Empirical Research.* Campus Verlag, Frankfurt, 2012.