Using E-Mail Social Network Analysis for Detecting Unauthorized Accounts

Adam J. O'Donnell*
adam@cloudmark.com
Cloudmark, Inc.
128 King St, 2nd Floor
San Francisco, CA, USA

Walter C. Mankowski
walt@cs.drexel.edu
CS Department
Drexel University
3141 Chestnut St.
Philadelphia, PA, USA

Jeff Abrahamson
jeffa@cs.drexel.edu
CS Department
Drexel University
3141 Chestnut St.
Philadelphia, PA, USA

ABSTRACT

In this paper we detail the use of e-mail social network analysis for the detection of security policy violations on computer systems. We begin by formalizing basic policies that derive from the expected social behavior of computer users. We then extract the social networks of three organizations by analyzing e-mail server logs collected over several months and apply the policies to the resultant social network and identify subsequent policy violators.

After closer examination of the outlier accounts, we find that a significant fraction of the suspect accounts were supposed to have been terminated long ago for a variety of reasons. Through the analysis and experiments presented in the paper, we conclude the analysis of social networks extracted from network logs can prove useful in a variety of traditionally hard to solve security problems, such as detecting insider threats.

1. INTRODUCTION

We human beings are social creatures. As we use our computational tools to communicate with one another, the social interactions which we engage in leave impressions in network traffic and log files. The map of communication that binds a community can be extracted from a variety of sources, such as network traffic traces, file shares, and IM logs. The typical daily usage seen by corporate and educational department e-mail servers, for example, generates predictable patterns in the social network that can be quantified using graph theory. Similarly, misuse patterns, such as the generation of traffic from accounts being controlled by unauthorized users, appear as anomalies in the social network which can also be easily quantified.

In this paper, we detail the use of e-mail analysis for detection of security violations on a computer system. We begin by translating the strictly enforced polices and weakly enforced social mores which govern the use of e-mail systems into the language of graph theory. The policies are then applied to e-mail social networks we have extracted from several months of mail transaction logs from multiple organizations. Upon closer examination of the users which violated the policies we defined, we find that a significant fraction of the suspect accounts were supposed to have been terminated long ago for a variety of reasons. Through the analysis and experiments presented in the paper, we show that:

- Social networks extracted from system log files can be useful in finding individuals who are causing security violations.
- Many properties of social networks that have an impact on system security are computationally feasible to evaluate, and in fact can be determined in linear time.

The architecture and techniques presented are not a replacement for the standard policy enforcement mechanisms which are necessary for controlling well-described security issues. Firewalls and access-control lists are critical for enforcing the strict security policies which are common in real-world computer networks. We contend, however, that policy enforcement can also be applied on months of social interaction data extracted from system logs. The system administrator can then act on any accounts identified as policy violators as he or she sees fit.

The social network analysis system proposed in this paper is an attempt to solve a specific instance of the malicious insider problem, which is known to be a notoriously difficult and critical issue in the security community. Our architecture is capable of identifying invalid accounts held by users who, on the surface, appear to be accepted members of an organization but do not exhibit the same social behaviors as others in the organization. The social network analysis system proposed in this paper may not identify all invalid accounts located on a server. If an account is held by an unauthorized individual who follows all of the social behaviors that we define to be acceptable, then our system would not flag these users for further analysis. Additionally, our architecture is not useful in finding invalid accounts on ISP-based mail servers, nor is it an effective method of spam control. In both instances there is no predictable social structure that emerges out of the logs which can be exploited for finding misbehaving accounts.

2. RELATED WORK

Our work can be viewed as an application of social network [1] analysis to information security. Techniques similar to those discussed in this paper are currently being applied to spam filtering and e-mail manageability, for example [5,

*Corresponding Author

CEAS 2006 - Third Conference on Email and Anti-Spam, July 27-28, 2006, Mountain View, California USA

7]. Tyler, Wilkinson, and Huberman showed it was possible to use graphs derived from e-mail to divine the organizational structure of a corporation [11]. Cortes, Pregibon, and Volinsky showed how telephone fraud can be detected by comparing the social behavior of new telephone accounts to that of previously tagged fraudulent accounts [3]. Using techniques which are in the same spirit as Cortes' work, we show that determining if an individual is participating in an organization through standard social norms via e-mail is an effective method of locating invalid accounts on a system.

It is possible to describe our work as an application of intrusion detection, or IDS, methodologies to e-mail social networks. For example, graph-based approaches in intrusion detection have been explored in [2] for network traffic analysis. Our described method is, in many ways, a combination of rule-based [9, 10] and anomaly-based IDS whose data set is the graph created by user interaction [4, 6]. Like rule-based IDS, the system uses rules explored in Section 3.1 to detect anomalous behavior. The analysis, however, must be done on a volume of accumulated data and is not directly applicable to online analysis. Unlike anomaly-based intrusion detection systems, no training time is required to begin the analysis. Anomaly-based techniques can be used after the rule application process to further filter out known acceptable behavior before presenting the report to the analyst. Time is required to accumulate enough logs to construct the graph defined by individual interactions; we provide an analysis of how much data was necessary from our data sets in Section 4.

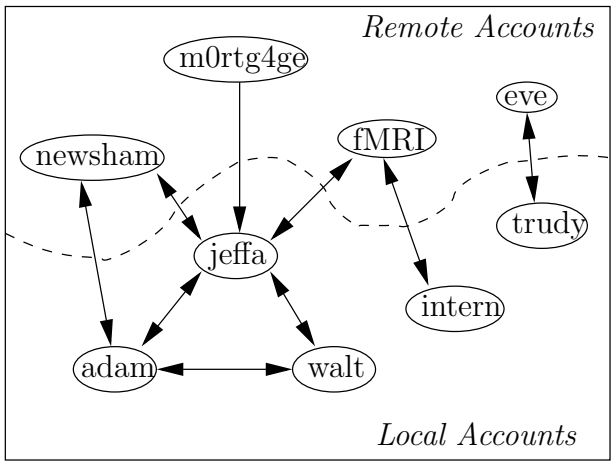

Figure 1: A sample subset of e-mail transactions is shown in the above figure. The majority of behavior is normal for a departmental e-mail server, where cliques form amongst the internal users and communication occurs with outside collaborators. The smaller disconnected component consisting of `eve` and `trudy` constitutes suspicious behavior due to its lack of interaction with the rest of the group, however.

3. POLICIES ON GRAPHS

We consider an example graph of interactions found on a departmental e-mail server in Figure 1. Users `adam`, `jeffa`, and `walt` interact with one another on a regular basis, sending e-mails back and forth. Outside collaborators, specifically `newsham` and `fMRI`, communicate with `jeffa` on a regular basis. While not a member of the larger community, `intern` shares an account on the same server and communicates with one of `jeffa`'s coauthors, `fMRI`. Certain users appear to be engaging in behaviors that are counter to the group's mores; one user, `m0rtg4ge`, is sending e-mails to `jeffa` which go unacknowledged. Users `eve` and `trudy` seem to be carrying on communication which is completely separate from the user community.

The rules of social behavior that exist are not completely arbitrary. We expect individuals who hold an account on a sever occupied solely by a single community to interact as if they were members of that community. People within the same community should communicate with one another. If a user is found not to collaborate at all with any other members of the community, however, the user may either be a rugged individualist or the user may be accessing an account illegitimately. While the former is perfectly acceptable behavior, the latter is clearly unacceptable. Therefore, it would be useful to write policies that describe properties of the user's social network and then catch behaviors which do not match these policies.

3.1 Example Policies and Algorithms

We begin defining our policies by first formalizing our e-mail topology. Let the graph which is generated by e-mail traffic be defined as $G = (V, E)$, with E being a set of directed edges where each edge represents the existence of at least one e-mail between two users. The subgraph induced by the e-mails observed by a given host s as $G_s = (V, E_s)$, where V is the set of e-mail users, $V_s \subseteq V$ is the set of e-mail users who have accounts on the host s, and $E_s \subseteq E$ is the set of edges which have at least one endpoint in V_s.

Our first statement is the relatively standard policy of disallowing relay e-mails, which is strictly enforced on many e-mail servers on the Internet. This policy, which we refer to as No-Relay, is a standard technique in spam control that requires that either the source of the e-mail or the destination of an e-mail message must be an account handled by the host.

We do expect each user of the e-mail server to communicate with at least one other user of the e-mail server over some period of time. While it is not mandatory for individuals to periodically check in with one another, it might be anomalous for an account holder not to communicate with any other account holder over some period of time. Specifically, the policy requires that every user on the system be an endpoint for a bidirectional edge, referred to as *reciprocity* in the literature [12].

In terms of e-mail interaction, the policy states that every user sends out at least one e-mail that is responded to by the recipient. The application of this policy has the added benefit of filtering out spam from our data set. In the example provided in Figure 1, the policy would remove `m0rtg4ge`'s messages from the social network.

Our second policy, named Collaborate, is a formalization of this policy. In graph theory terms, there will exist a bidirectional edge walk from every vertex to every other vertex on the graph for all vertices in V_s. A useful, but weaker, form of this policy is called Weak-Collaborate, which states individuals are in the same collaboration chain if each communicating pair is on the same server or if they share an outside collaborator. In other words, a walk is accepted by the Weak-Collaborate policy if every vertex

```
# G contains the e-mail graph
# G_c contains the "collaborators"
# G_wc contains the "weak collaborators"
# V(G), E(G) denote the vertex and edge sets of a graph G.

Main:
    Initialize G ← (V ← ∅, E ← ∅)
    Initialize G_c ← (V_c ← ∅, E_c ← ∅)
    Initialize G_wc ← (V_wc ← ∅, E_wc ← ∅)
    Load e-mail log files
    Load all accounts under our control into V_s

    for each log entry:
        parse {v_source, v_dest}
        if {v_source, v_dest} ∉ E:
            if v_source ∉ V:
                V ← V ∪ v_source
            if v_dest ∉ V:
                V ← V ∪ v_dest
            E ← E ∪ {v_source, v_dest}

    for each {v_source, v_dest} ∈ E:
        if {v_dest, v_source} ∈ E:
            if v_source ∈ V_s ∧ v_dest ∈ V_s:
                E_c ← E_c ∪ {v_source, v_dest}
            if v_source ∈ V_s ∨ v_dest ∈ V_s:
                E_wc ← E_wc ∪ {v_source, v_dest}

    CPolicyBreakers ← V_s −
        LargestConnectedComponent(G_c)
    WCPolicyBreakers ← V_s −
        LargestConnectedComponent(G_wc)

    Return CPolicyBreakers, WCPolicyBreakers
```

Figure 2: Pseudo-Code Implementation of the Policy Violation Detector.

in the walk contained in V_s is separated from its neighboring vertices contained in V_s and on the walk by at most one vertex not in V_s.

We therefore can define a policy violator as any account situated on the server which does not lie in the largest connected component of the graph, which allows us to evaluate these policies in polynomial time. In Figure 2 we present an algorithm for identifying users responsible for violating the specified policies. The algorithm first selects the edges critical to the graph as defined by the COLLABORATE and WEAK-COLLABORATE polices. The largest connected component is identified, and any vertices which lie outside the largest connected component and exist on the mail server are reported to the administrator. After further examination, the administrator can decide if the account is invalid, and if so, deactivate the account.

4. EXPERIMENTAL RESULTS

Our initial data set was generated from `ece.drexel.edu`'s mail logs, and consisted of 1,038,939 log entries for each e-mail sent and received from January to September 2003 by the domain's `sendmail` server. When examined as a whole, the logs contain 337,773 unique {to, from} e-mail address pairs generated by 251,348 unique accounts, the majority of the mail transactions being due to spam. The resultant graph is not fully connected and contains 50 disjoint components. We create a new graph, which we refer to as the *reciprocity subgraph*, by restricting the edge set to include only bidirectional edges. The resulting graph is reduced to 37,618 {to, from} address pairs, or 18,809 undirected edges. These edges exist between 12,408 vertices, which are now unique e-mail ID's, in 14 separate connected components, with the largest connected component consisting of 12,354 vertices and 18,768 undirected edges. These statistics are summarized in Table 1.

	$\|V\|$	$\|E\|$	CC Count
Raw Data	251,348	337,773	50
Reciprocity Subgraph	12,408	18,809	14

Table 1: The above table provides a summary of the number of vertices and edges present one of the author's department's e-mail logs. The graph dramatically reduces in size after removing all edges which do not have a reverse edge and all vertices which do not have any incident bidirectional edges.

		Violations	
Policy Tested	Examined	Suspect	Confirm
COLLABORATE	294	37	14
WEAK-COLLABORATE	294	13	8

Table 2: The COLLABORATE and WEAK-COLLABORATE policies were applied to the reciprocity subgraph, which then uncovered 36 and 13 possible policy violators out of 294 active accounts, respectively. After discussion with the system administrator, it was decided that 14 of the accounts found by the COLLABORATE policy should be immediately terminated. The confirmed violators of the WEAK-COLLABORATE policy formed a subset of the confirmed violators of the COLLABORATE policy.

The reciprocity subgraph is then further processed using the algorithms presented in Section 3.1. As shown in Table 2, out of a possible 294 active accounts, 37 suspicious accounts were found using the COLLABORATE policy, and 14 suspicious accounts were found using the WEAK-COLLABORATE policy. When presented with the latter policy violators, the `ece.drexel.edu` system administrator found that 8 should have been terminated long ago, with the remainder of the accounts being owned by webmail users and one deceased professor with an auto-responder placed in his account. After being shown the remainder of the accounts found by the COLLABORATE policy, an additional 6 accounts were found that were clearly illegitimate.

It was not possible for the authors to gather statistics on the total number of false negatives, or users who were actually invalid but not labeled as such by our architecture. Our system administrators, while extremely accommodating, did not have the resources to validate every account on the system.

4.1 Required Depth of Data

While it would be more efficient from both a computational and a log storage aspect to examine only a subset of the e-mail logs to detect policy violators, as can be seen in Figure 3 this may not always be possible. Invalid account holders may rarely communicate with other individuals off the system, necessitating the use of deep communication logs. Additionally, while the majority of e-mail account holders fall into a single community after a short period of time, the number of individuals who join this community continues to increase over the course of several months.

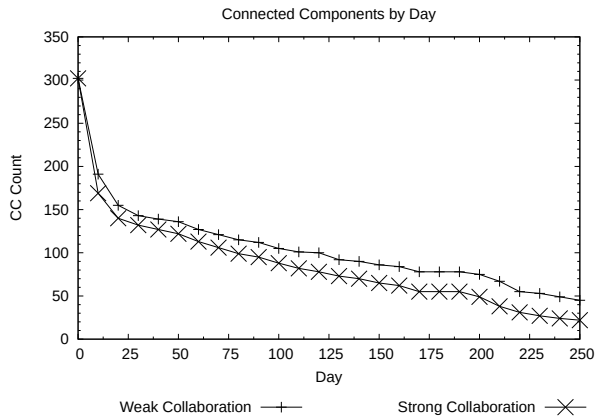

Figure 3: We examine the volume of logs required to perform our analysis by counting the number of connected components present as edges are added to the set of e-mail users who are active over the log usage period. While the majority of users end up in the same connected component after one month, the number of connected components continues to slowly decrease over nine months.

Therefore, not only will examining lengthy portions of logs increase the number of policy violators that can be detected, it will also decrease the number of false positives, or individuals who are incorrectly tagged as being social outliers, found by our system.

The authors do believe that a windowing scheme, such as the one presented by Cortes in [3], would be useful for combatting a social engineering attack mounted by holders of invalid accounts. In the attack, invalid account holders would occasionally attempt to convince a valid account holder to communicate with the attacker. If successful, the invalid account holder would place themselves inside the main connected component and remove themselves from consideration by our system. To combat this attack, a windowing system can be used to remove edges temporarily from the largest connected component, which would expose components which are only loosely connected to the larger social network.

Preliminary analysis of five contiguous months of mail logs from two additional sources, namely logs from `cs.drexel.edu` and `ece.vill.edu`, shows similar disconnected groups of users. Due to confidentiality restrictions on the data, we are not able to fully quantify which accounts are invalid beyond a confirmation from the mail administrators that we have identified several problematic accounts on their systems.

5. CONCLUDING REMARKS

E-mail server log data from communities of people who are expected to know each other (academic department servers, university servers, internal corporate servers) provide a rich trove of social network information. Using graph theoretic techniques, especially analyzing the connected components of the weak and strong conversation graphs, is highly effective at generating candidate intruders.

While we concentrate on e-mail interaction graphs in this work, our techniques are not limited to e-mail communication analysis. For example, the access patterns of documents on a shared file system can be used as a basis for construction of a social interaction graph. While the traffic traces associated with client-server file shares encode only client-server communication, they also implicitly encode a relationship between humans on the clients and documents located on the servers. The links from authors to documents give rise to a bipartite graph, where the bipartite graph's one mode projection [8, 12] of the document authors can then be analyzed using techniques similar to those presented in this paper.

In future work, we plan on examining the forms of attacks which can be applied against our architecture, and the corresponding countermeasures which can be employed against these attacks. For example, the social engineering attack mentioned in Section 4.1 may be combatted through the use of the mentioned windowing system or by examining only k-connected components as opposed to singly connected components.

Acknowledgments

The authors gratefully thank Jonathan Hoult of Drexel's ECE department, Gaylord Holder of Drexel's CS Department, and Rick Perry of Villanova University for kindly providing SMTP logs. The authors also extend their thanks to Eric Cronin and Tim Newsham for their early reviews of this work.

6. REFERENCES

[1] A. L. Barabási. *Linked: The New Science of Networks*. Perseus Publications, 2001.

[2] S. S. Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS: A graph based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, 1996.

[3] C. Cortes, D. Pregibon, and C. T. Volinsky. Communities of interest. *Intelligent Data Analysis*, 6(3):211–219, 2002.

[4] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for Unix processes. In *Proceedings of the 1996 Symposium on Research in Security*, pages 120–128. IEEE Computer Society Press, May 1996.

[5] J. Golbeck and J. Hendler. Reputation network analysis for email filtering. In *Proceedings of the First Conference on Email and Anti-Spam*, Mountain View, CA, July 2004.

[6] W. Lee and S. J. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, January 1998.

[7] C. Neustaedter, A. J. B. Brush, M. A. Smith, and D. Fisher. The social network and relationship finder: Social sorting for email triage. In *Proceedings of the Second Conference on Email and Anti-Spam*, Mountain View, CA, July 2005.

[8] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(026118), 2001.

[9] V. Paxson. Bro: A system for detecting network

intruders in real-time. *Computer Networks*, 31(23–24):2435–2463, December 1999.

[10] M. Roesch. Snort: Lightweight intrusion detection for networks. In *Proceedings of the 13th Systems Administration Conference (LISA)*, 1999.

[11] J. R. Tyler, D. M. Wilkinson, and B. A. Huberman. Email as spectroscopy: Automated discovery of community structure within organizations. In *Communities and Technologies*, pages 81–96. Kluwer, B.V., 2003.

[12] S. Wasserman and K. Faust. *Social Network Analysis*. Cambridge, 1994.