

SPECIAL ISSUE PAPER

Virtual flow-net for accountability and forensics of computer and network systems

Daisuke Takahashi, Yang Xiao* and Ke Meng

Department of Computer Science, The University of Alabama, 101 Houser Hall, Box 870290, Tuscaloosa, AL 35487-0290, U.S.A.

ABSTRACT

Information/secret leaking cannot always be recorded in digital log files. In other words, in log files, not all information/events are recorded, and it is thus impossible to trace the paths of secret leaking on the basis of log files alone. In this paper, to resolve the difficulty of the lack of information, we utilize user–relationship graphs, or social networks, to compensate for the required information. We also utilize a probabilistic analysis to build virtual links to follow information flows. User–relationship graphs are constructed from several flow-net data structures over a longer period so that we can avoid missing embedded threats such as hostile codes. We call this approach virtual flow-net. Copyright © 2011 John Wiley & Sons, Ltd.

KEYWORDS

social networks; user–relationship graph; flow-net; accountability; forensics

*Correspondence

Professor Yang Xiao, Department of Computer Science, The University of Alabama, 101 Houser Hall, Box 870290, Tuscaloosa, AL 35487-0290, U.S.A.

E-mail: yangxiao@ieee.org

1. INTRODUCTION

Computer forensics, or digital forensics, is an emerging technique used for the methodical investigation of computers and digital devices to solve criminal cases. In the 20th century, the rise of personal computers and computer networks made people's lives convenient, but it also increased the number of crimes involving them. Accordingly, some people have intentionally misused computers to commit crimes or to attack other computers [1]. Digital crimes include hostile codes (e.g., computer viruses and worms), spam e-mails, denial-of-service attacks, and so on. However, involving computers in crimes often leaves electronic evidence in the computers. In fact, even though criminals believe all relevant information in the digital devices has been deleted or overwritten by other irrelevant data, actual traces remain in the hard disk drives (e.g., in slack space). Sometimes, written data cannot be deleted but instead are only marked as having been deleted, and space is free to be reallocated [1]. Furthermore, by using sophisticated electron microscopes, even overwritten tracks can be recovered in a bit-by-bit manner [1]. Basically, computer forensic specialists reveal and show the digital evidence in a court and then collect segments of data, extracting the meanings of the data and user's activities that initially appeared to be meaningless at first glance.

From the traces left by criminals, detectives and computer forensic specialists figure out critical evidence of the crimes.

These investigations involve two kinds of methodologies: hard disk drive and auditing log file investigations. In hard disk drive investigations, the data examined by the digital forensics specialists reveals data that include contents that were hidden by the criminals. On the other hand, in auditing log file investigations, the data examined may not contain the contents but just discloses activities that were electronically conducted by the criminals. Both investigations are very important. However, in this paper, our primary concerns mostly address the auditing log files; that is, we are only concerned with users' activities and determining the meaning of the sequence of the activities indirectly. Indeed, only retrieving the user's malicious activities from the auditing log files is very challenging, but the use of these techniques is very practical in criminal investigations.

In our previous paper [2], we presented algorithms that use auditing log files to trace the possible paths of secret leaking with or without a bounded period to perform the following tasks: (i) to find out a possible route/trace on which a secret flows into a particular resource; (ii) to find out all the suspects who possibly accessed, directly or indirectly, the secret in the past or in the past time duration T ; (iii) to find out all possible routes/traces through which a secret flowed into a particular resource, and so on.

However, information/secret leaking cannot always be recorded in digital log files. Examples include the following: (i) a user can come home and tell the secret to his wife who

may eventually leak the secret to other places; (ii) the user can use other means (such as telephone, fax, cellular phone, etc.) than computers to leak the secret, which do not log the actions or cannot be accessed or associated with. In other words, in log files, not all information/events are recorded, so it is impossible to trace the paths of secret leaking based only on log files.

In this paper, to resolve the difficulty of the lack of information described earlier, we utilize user-relationship graphs (URGs), or social networks, to compensate for the required information. We also utilize a probabilistic analysis to build virtual links to follow information flows. URGs are constructed from several flow-net data structures over a longer period in order to avoid missing embedded threats such as hostile codes. Of course, the length of the information durability is another research issue, but in this paper, we only use 1 year of information for the construction of URGs. We call this approach virtual flow-net.

In this paper, we use information leakage as an example to illustrate usage for forensics. However, virtual flow-net does not limit itself in the usage of information leakage. The idea could be used in many situations when some information is missing. This idea potentially improves the effectiveness of forensics.

One major contribution of the paper is the adoption of URG in flow-net or logs so that missing links can be built virtually.

The rest of this paper is organized as follows: Section 2 introduces social networks and k -connect graphs; Section 3 reviews auditing log files and flow-net; Section 4 proposes the virtual flow-net; Section 5 explains how to construct a URG and an extended URG (EURG) and shows algorithms; Section 6 provides some experiments; Section 7 presents some problems and challenges in URG; finally, we conclude the paper in Section 8.

2. SOCIAL NETWORK AND K -CONNECTED GRAPH

Part of the process of constructing a URG utilizes the concept of social networks and the k -connected graph. Actually, a URG is a social network constructed from particular data sets (e.g., server side's e-mail log files, system log files, driver license database, etc.). The concept of the k -connected graph, or k -connectivity, is used to put weight on links among people in the social network. By using these two concepts, we construct virtual links compensating for missing links to figure out virtual paths of secret leaking.

2.1. Social network

Broadly, a social network is an actual network of people all over the world. It also examines the small-world phenomenon. In general, the concept of a social network is used to figure out short paths between two people in a network from only local information [3]. For example, when people seek jobs, one way to attain job interviews might be to follow human

connections until they finally reach representatives of their target jobs. This strategy might work better than more direct ones. However, how do people determine short or efficient connections to attain their target jobs? People usually only have the information of their immediate contacts, and they usually do not have global information for their social networks. Thus, the question confronted within a social network is how to determine short or efficient paths for people to attain their target jobs in a social network from only their immediate contacts in this job search example.

A social network consists of clusters of smaller subnets, and usually, these clusters are hierarchically structured. Individuals belong to one or more social subnets and have relationships with others in their networks. Some individuals sometimes do liaison work between or among subnets that connect two or more unrelated social networks. The smallest unit is the individual, and each may have one or more immediate contacts, which construct the first levels of social networks. Obviously, each participant in this first level has their own social networks; therefore, including their contacts makes the second-level social networks, that is to say, contacts-of-contacts. These processes are recursively conducted until the social networks finally cover all the people on earth. Therefore, in approximation, if each individual has 1000 immediate contacts or acquaintances, 1000^3 or one billion people are involved in an individual social network before its third level (contacts-of-contacts-of-contacts) and covers everyone in the USA [3]. That means that only three consecutive links are required to get to anyone in the USA. Of course, this result is subject to an assumption of the networks being random; that is, no one belonging to a network has the same person in their contacts. In other words, there are no overlaps among acquaintances of two people, but this is not true in reality. However, in fact, social networks are constructed differently according to their original resources. For example, social networks constructed by e-mail contacts and online communities, such as Club Nexus or Facebook, may have different sizes and structures. The authors in [3] found out that there are some students participating in Club Nexus who do not have any links to other students, and therefore, these people registered themselves to the community but did not register friends' links, called 'buddies', to it. On the other hand, these people may have friends' e-mail addresses in their address books in e-mail client applications.

Accordingly, because social networks grow exponentially according to the number of average immediate contacts, targets are not relatively far from the original person but just a few steps away from them [3]. In the previous example, everyone in the USA is virtually covered by only three links. This phenomenon is called the small-world phenomenon. To reveal the small-world phenomenon, a number of experiments and studies were conducted since the middle of the 1960s [4–8]. For example, an experiment of e-mail contacts with 60 000 participants conducted in 2002 demonstrated an average of 4.1 links chained people in continents end-to-end [9].

In searching for short paths between two people, two organizations, or a person and organization from many

contacts, greedy algorithms are often used [3]. This is because, in most of the cases, individuals only have knowledge of local contacts, which may be, at most, knowledge of contacts-of-contacts or even just contacts. However, paper [3] experimentally proves that even this kind of information is enough to connect originated people to targets overseas by using greedy algorithms with auxiliary information such as geographical or professional proximity. In other words, every time a person chooses one or more contacts from their own contact list as a next step, by choosing ones who have geographical or professional proximity, the targets are successfully reached in very few steps.

Another assumption of the small-world phenomenon is that all networks are subnets of one or more larger networks, which means that a social network is hierarchically structured or tree shaped, as shown in Figure 1 [10]. For example, we can say that a student belongs to a research laboratory within the Department of Computer Science, which belongs to either the College of Engineering or Arts and Sciences, which belongs to some University, and so on. We can also say that this student may be an undergraduate student who belongs to some local society or community. In this hierarchically structured social network, we can assume that two individuals belonging to the same group or organization have more probability of knowing each other than ones belonging to different groups or organizations, and perhaps, connections are stronger, or more intimate, than ones in different groups [3]. Paper [3] raises a probabilistic analysis of the strength of these connections such that the probability that two individuals know each other can be written as $e^{-\alpha h}$, where h is the height of the tree at their lowest common branch and α is the decay parameter. Typically, paper [3]

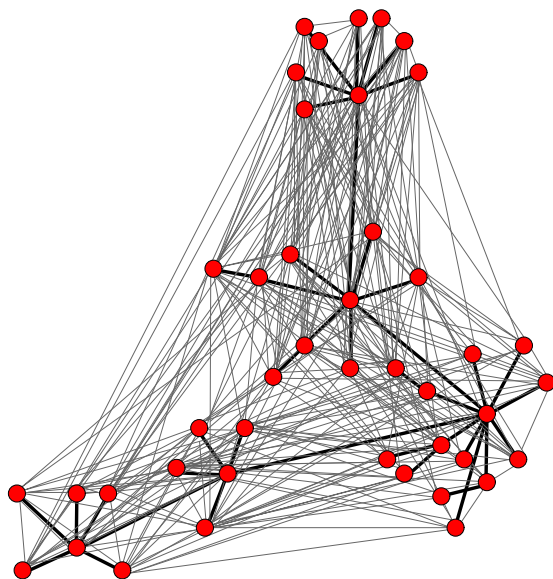


Figure 1. An example of a social network constructed according to e-mail communications.

weighs the closeness of two people by $-\alpha$ so that two people in the closer two groups are more likely to know each other, as we described previously.

It was proven by Kleinberg [11,12] that by using the geographical closeness to targets in each hop (each contact), the lengths of contact chains can be bounded by $(\ln N)^2$ by using a simple greedy strategy, where N is the number of people in a social network. In other words, whenever people select contacts from their lists, choosing a next contact that is geographically closer to the targets' location causes the length of the contact chains to be bounded by $(\ln N)^2$. However, to achieve this upper bound of $(\ln N)^2$, Kleinberg [11,12] also made the assumption that the probability that two individuals known each other must be the inverse of the square of the distance; that is to say, when the distance of two individuals is d , the probability that they both know each other can be represented as c/d^2 , where c is some constant. Otherwise, it is impossible for a person to greedily find out paths to the targets in polylogarithmic time [11,12].

2.2. E-mail network

Previously, HP Labs [3] constructed social networks from their e-mail logs. The construction of a social network follows the following rules: (i) two individuals are connected if they have exchanged e-mails at least six times both ways within 3 months; (ii) someone multicasting an e-mail to more than 10 people at the same time must be ignored.

A threshold of six e-mails may seem too weak to connect two individuals, but it is proven by Granovetter [13] that weak links can be used for job searching and information spreading.

As a result, an HP Labs experiment created a social network of 430 individuals, where the median number of acquaintances is 10 and their mean value of them is 12.9 [3]. From this social network, HP Labs investigate greedy searches according to three different properties: best connected, closest to the target in the organization, and closest physical proximity to the target. In the first strategy (i.e., best connected), contacts having the most acquaintances are chosen as the next hops. This strategy is best suited for power-law degree distributions, but in the filtered HP Labs e-mail network, distribution had an exponential tail rather than a power-law distribution. Accordingly, generated results had a median number of 16 and a mean value of 43 steps [3]. On the other hand, the second and third strategies yielded better results. In the second strategy, next contacts were chosen on the basis of their closeness to the targets in a hierarchically structured social network, generating a median of 4 and mean of 5 [3]. In the third strategy, as in the second one, contacts were chosen on the basis of their closeness to the targets, but with geographical rather than organizational hierarchy, yielding a median of 6 and a mean of 12 steps [3].

2.3. k -Connected graph

Basically, if there are paths that link any pair of vertices in graph G , G is considered to be connected. Otherwise, it is

disconnected if there is no way to get to one or more vertices from the other vertices in G . For example, the left side graph in Figure 2 is disconnected because it is obvious that there is a vertex that cannot be reached from the other five vertices, whereas the right side graph is connected because there is a path that links each pair of vertices.

Thus, k -connected graphs have k distinct (independent) paths for every pair of vertices in graph G . In other words, it is required that at least k vertices must be removed from the graph in order to disconnect it. In other words, removal of any $k - 1$ vertices does not affect the connectivity of the graph.

It is very important to figure out the number of connectivity of a graph because knowing the connectivity of the graph allows the robustness of network communication links to be interpreted. For example, when we represent a computer network as a connected graph and the graph is a minimum spanning tree, this computer network is very vulnerable to communication failure because the removal of only one edge disconnects one or more vertices from the graph (one or more host machines from the computer network). To avoid this vulnerability, a computer network should have one or more redundant links for every pair of host machines, routers, or switching devices. Thus, when a computer network is expressed as a graph and we know that the graph is k -connected, the computer network can be considered to be very robust because any $k - 1$ link failures do not cause total failure of the computer network. One problem of the k -connected graph is that whether or not a graph is k -connected cannot be computed by any algorithms in polynomial time. In other words, figuring out whether a graph is k -connected or not is known as a non-deterministic polynomial-time hard problem [14]. Fortunately, several approximation algorithms for the k -connected graph problem have been developed and turned out to have the polynomial time complexity [14]. For example, for the unweighted k -edge-connectivity problem, an approximation algorithm with a performance ratio of 1.85 has been developed [14].

We can apply this notation of k -connected graphs to express the strength of the network, which is the probability that two individuals in a social network know each other.

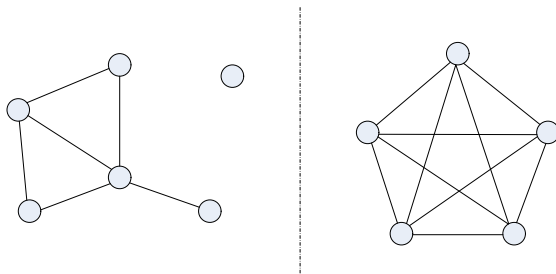


Figure 2. Two graph examples.

2.4. Related work

Related work in accountability includes those in [1,15–31]. Other related work can be found in [32–81].

3. AUDITING LOG FILES AND FLOW-NET

3.1. Auditing log files

Typically, auditing log files aim to record every activity of users involved in computer systems or networks. These data are usually ordered in a time-sequential manner and preserved on hard disk drives. Later, they are retrieved by investigators, such as detectives or system and network administrators, so as to produce scientific evidence of crimes. This is done because, in quite a few cases, electronic documentary evidence is eliminated from electronic storage, such as hard disk drives, by clever offenders (although it is almost impossible to eliminate digital evidence completely and they are typically left in slack space). From the log files, even though electronic documents have already been eliminated, detectives or system and network administrators may find clues or doubtful activities through which they can identify and finally relate the criminals to the cases.

In general, many resources apart from operating systems have their own log files. For example, e-mail client applications obviously have their own log files, which record logs of sent and received messages, which determine who sent an e-mail to whom and when? Additionally, printers in a LAN should have logs of who made use of them and when? In other words, they observe all accesses from local hosts and keep them in their own log files. Along with these application log files, the network traffic or packets in networks can also be captured and logged into files in particular formats. For example, tools used for this purpose are *ethereal*, *wireshark*, *tcpdump*, and so on.

In our previous papers [2,82], we assumed that log files can keep all the users' activities in terms of user or process names, actions, resources, and possibly locations with time-stamps. In short, they keep things such as who does what to whom and when. Basically, user or process names represent instances, which initiate events. They usually include user and process IDs and IP addresses. Actions keep users' behaviors, such as reading or writing in database systems or sending e-mails in network systems. Resources are objects through which actions are carried out by users, and they include files, directories, servers, and so on.

In this paper, we consider two kinds of log files with respect to whether users interact with other users, file systems, or resources. The former case comprises e-mail log files in e-mail servers of Web-based e-mail applications. In general, e-mail servers keep logs of exchanged e-mails, such as the sender's user accounts, the recipient's e-mail addresses, the subject, and the time, which look like as follows:

Outbox for user D:

- User A, Hello, Feb 4, 2008
- User B, Re: Hi, Feb 4, 2008
-
- user-c@example.com, Fw: Pictures, Feb 3, 2008
-

We assume that this kind of log file only records the users' activities but not the exact contents of logs. In the aforementioned example, a log file keeps the fact that user D sent an e-mail to user B on 4 February 2008, but it does not keep the contents of the e-mail. On the other hand, investigators, such as detectives or system and network administrators, can prove relations between users A and D from the e-mail log file. Although this information may be trivial in this instance, it may convey useful information for later investigation. Therefore, these log events will typically be used to prove relations between two people. Moreover, we cannot apply a transit trusted model to create URGs, or social networks, because we cannot always say that two people have a connection even when these two people sent e-mails to the same third person.

In the latter case, log files should log information as follows.

- User A wrote file X at 10:05 AM, 4 February 2008 (1);
- User A accessed server Y at 10:10 AM, 4 February 2008;
- ...;
- User B read file X at 04:30 PM, 4 February 2008 (2);
- User B deleted file X at 04:32 PM, 4 February 2008;
-

Note that the logging in this system is just for an example to illustrate our idea. Logging in other systems also has similar features.

This type of log records is called system logs and is mainly used to disclose data or file manipulation or elimination and for rollback and roll forward transactions recovered from accidental crashes in the database systems. From this log information, if investigators become suspicious of user B's behavior, they may, if possible, recover the data in file X and determine its contents. Thus, from these log data, it is shown that there was a file X, which was deleted by user B, and it may be worthwhile to investigate this file for more clues. Log files also reveal other information, such as indirect relations between two people. Indirect relations are two people's relations that are not explicitly represented in data but are semantically induced from their contents. For example, from the aforementioned logged data, user A made some modification to the data in file X at 10:05 AM (1), and at 04:30 pm on the same day, user B may have seen this modification by reading (2). In this case, we may be able to create some weak relations between users A and B. Thus, as in the previous case, it may convey useful information for the later investigation.

3.2. Problems in log files

One problem of these log file systems is that they are not organized well, which results in confusing investigators.

Because, in most of the cases, auditing log files are ordered in a time-sequential manner, two consecutive logs or actions are likely to be irrelevant or not related to each other. This can be especially true for the network auditing log files. Moreover, in the network auditing log files, which are captured by the aforementioned network monitoring tools, even one action produces many packets. For example, when a user attempts to send 10 MB of attached data via e-mail or FTP (e.g., pictures), these data are usually divided into a number of smaller chunks and separately sent to one or more recipients. Consequently, these chunks are assembled by the recipient side of an e-mail client application. In this situation, network monitoring tools capture the same number of packets sent by the sender representing the same actions for a number of times, but these actions may not be sequentially logged by the tools and thus may confuse investigators.

Another problem of auditing log files is that different log files are saved in separate files so that the relationship among events in different log files is missing.

3.3. Flow-net

In previous papers [15,16,83,84], in order to mitigate their mess and make log files more manageable, we designed another type of data structure, called flow-net, which was especially suited for retrieving the activities of particular individuals.

Figure 3 shows an abstract image of flow-net, in which there are four users and four resources in connection, which interact with each other. For example, from the figure, we can know that user 1 accessed resource A at 10:00 AM and resource D at 2:00 PM. We can also know that user 2 accessed resource B at 12:00 PM. On the other hand, resource A is accessed by user 1 at 10:00 AM and by user 4 at 11:00 AM. However, actual flow-nets can be more complicated. Figure 3 shows that a flow-net is a two-dimensional linked list with two aspects: users and resources. However, flow-net can have three dimensions, as indicated in our previous papers [15,83,84].

In flow-net, every user's activity is organized both user-by-user and resource-by-resource by creating a number of linked lists, each of which corresponds to an individual and where the same resources are connected to each other in a time-sequential manner. In other words, we create linked lists according to all entities in the system, such as user names, user IDs, process IDs, and so on, so that each user has his or her own linked list (a sequence of activities). After constructing all the users' activity linked lists, the same resources of the linked lists are connected in the same way to create another kind of linked list, but at this time, activities are linked in terms of having the same resources. Therefore, a user activity or event, or a node of linked lists, has two incoming edges and two outgoing edges. In other words, one outgoing edge points to the same user's next activity and the other points to the same resource's next activity.

One advantage of flow-nets is that less time is required to search one user's activities. Because log files of multiple users' activities can be seen as sequences of each individual's activities, if it is required to look for some part of only a few users'

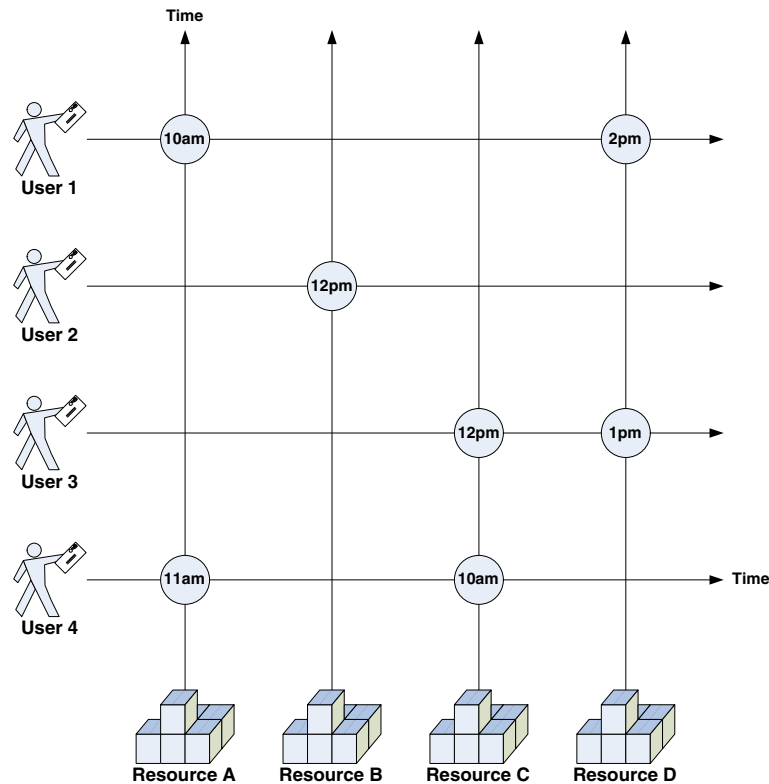


Figure 3. A simple image of flow-net. Human shaped figures on the Y-axis represent user accounts or processes in a network (auditing log file), and block shaped figures on the X-axis represent resources that any user can access. Also, circled intersections with times of each axis represent activities that users conduct processes. For example, user 1 accessed resource A at 10:00 AM, accessed resource D at 2:00 PM, and so on.

activities, they require less searching time than tracing all the log files from top to bottom. Another advantage of flow-nets is in network auditing log files; because individuals' activities are organized with respect to each user, one user's successive activities are captured more easily than the original log files. In other words, in general, one user's consecutive actions or packets are comprised of some relational activities, such as 10 MB of pictures being sent in a number of smaller chunks of data, so that we can automatically or manually assemble these chunks of data to construct other relational information, which creates linked lists comparatively easily.

A final advantage of flow-net is that it records the relationships of events.

Our previous works [15,16,83,84] studied flow-net. However, this paper is different from our previous works by studying virtual flow-net using URG.

4. VIRTUAL FLOW-NET

4.1. Multi-level access control and covert channels

With respect to database security, multi-level access control or role-based access control is typically employed in order to

preserve data confidentiality. In other words, each user must belong to one of the groups having their own access privileges, on the basis of which they can access various resources. For example, each datum may be classified into four privileges: top secret, secret, classified, and unclassified. According to the security classifications, the Bell-LaPadulla properties [85] restrict data access for the sake of preserving data confidentiality (i.e., no-read-up and no-write-down). With the no-read-up property, lower-level users must not access (read) any datum with higher privileges; with the no-write-down property, users with higher privileges must not write any datum to lower resources because this datum can be consequently read by lower-level users. However, paper [85] suggests that this may not be sufficient to prevent covert channels.

On covert channels, data with higher access permission may be indirectly passed to lower-level resources (files). Accordingly, users with lower-level privileges can access data with higher access permissions flowed via the covert channels. However, covert channels indeed violate confidentiality of data-based systems, even those with multi-level or role-based access control property. Furthermore, these covert channels can occur intentionally or unintentionally.

In papers [2,82], we presented the use of auditing log files to figure out possible paths (covert channels) of data flow from higher to lower levels when such paths occur.

We discovered that auditing log files could be represented by a specific graph notation, which is more like a binary tree. We designed algorithms on the basis of the depth-first search of the graph theory to reveal the covert channels and user accounts concerned with data leakages [2,82]. Our simulation results showed that the algorithms only need to address linear time complexities to figure out all possible user accounts that are likely to be on the covert channels.

4.2. Virtual flow-nets

However, information/secret leaking cannot always be recorded in digital log files. Examples include the following: (i) a user can come home and tell the secret to his wife, who may eventually leak the secret to other places; (ii) the user can use other means (such as telephone, fax, cellular phone, etc.) than computers to leak the secret, which do not log the actions or cannot be accessed or associated with. In other words, in log files, not all information/events are recorded, so it is impossible to trace the paths of secret leaking on the basis of log files alone.

Our supposition is that, even though we know that there must be data leakage via one or more covert channels to lower-privileged users, we cannot determine the paths from an existing log file or flow-net information. In other words, even though we know that some users with lower access permissions could obtain data from higher-level resources, our designed algorithms could not figure out any link between a log datum where a secret was first accessed and one at which the secret was first read by a lower-level user, or returned false eventually. In that case, it is supposed that the auditing log database manager may have missed recording one or more auditing data, secrets were in-electronically passed to some person intentionally, or some logs were intentionally manipulated. However, the lack of data may compromise further investigations of this kind. Thus, we require auxiliary information or social relations of all the users registered in the database system to compensate for the lack of information. We create a graph data structure representing individual user relations to others, called a URG, from the past log files or flow-nets.

Thus, further investigation follows the steps below:

- (1) Create a URG from past logs of applications (e.g., e-mail client application).
- (2) Create an EURG from past system log files or flow-nets.
- (3) Compute weight/probability of links of EURG.
- (4) Add the EURG to current flow-nets (called virtual flow-nets) for investigation.
- (5) Examine the virtual flow-nets utilizing virtual links with the highest to the lowest probability until useful information is extracted.

In step 1, from information of the flow-net of an e-mail log file of the past 1 year (this period can be determined by the experienced investigators), users are directly connected to form a URG, or a social network. For example, if user A sent e-mails to user B within the past year, these two people

are connected in a URG. Thus, constructing a URG is not that difficult. To the constructed URG, in step 2, we need to add extra information (connections) from the system log files or flow-nets. As mentioned before, system log files or flow-nets may not show direct personal connections between two people, but only the activities of each person. However, as we dealt with information flows in papers [15,83,84], we can make connections among people by using the resources or files in a system. For example, when users A and B share some file X in some organization system, these two users can be connected via this intermediate file, even though they have not disclosed any direct contacts in the past. From this kind of information, we can create another kind of URG, the EURG.

The EURG is very useful. For example, in a short period, detectives and digital forensic analysts may miss the connections among particular people, even though they made actual contact before and after this period. In general, a short period does not seem long enough to contain the complete relations among people in a social network because of the shortness of time. On the other hand, involving very old information in a URG may result in more positive false results and, accordingly, impede the investigations because involving more information requires more searches with increased complexity. Therefore, one of the issues must be what length of time we need to use to construct a complete or approximately complete social network among people in the organization, and there is obviously a trade-off between the length of log files or flow-nets and the accuracy of the URG. The period that digital forensic analysts need to construct a URG depends on the experience of the analysts and is out of scope of this paper.

After the construction of the EURG, we next need to weigh each link or connection between two people according to the strength of the probability that the two people know each other. This strength is determined in relation to the number of e-mails exchanged between two people. In other words, the more e-mails that are exchanged between two people, the stronger we consider their connection.

5. USER-RELATIONSHIP GRAPH

When creating a URG, we take two steps to create two different graphs: a URG and an EURG. The objective of the first step is to construct a user-relationship graph, and the second step constructs an extended user-relationship graph. A main difference between the two graphs is that the first graph concerns direct connections among people in a network, whereas the second graph concerns indirect connections among people. For example, direct connections among people include a parents-children connection, an e-mail connection, a social network connection, and so on. Furthermore, when we say direct connections in a social network, such as Club Nexus or Facebook, we mean links posted on an individual introduction's page that leads you to their intimate friends' introduction pages. Thus, connections of these links construct a social network, as mentioned in the previous section, and we call this network a URG.

The EURG adds another aspect to the original URG. An EURG extends their relations to indirect relations. In the indirect relationship, connected people may never have met in person, but are connected through some resources. For example, database logs keep the activities of log-in users toward data in the system. They keep the information that one user interacted with one file at some point, and after that another user interacted with the same file at another point. In the case of the EURG, these two users are capable of being connected to each other even though they belong to different departments in an organization and have not actually met each other. This is because, from our perspective, they had an interaction through the file and it is possible that the latter user saw data that had been written by the former user. We call this an indirect connection.

5.1. Creating user-relationship graph from a flow-net

In the previous subsection, we showed what log files look like and how they work in computer forensics. In this subsection, we show how to create URGs from these log files. As previously mentioned, creating URGs takes two steps, and these steps create two kinds of related graphs: URGs and EURGs. Now we go into detail about these two steps.

5.1.1. Step 1: creating a user-relationship graph.

Figures 4 and 5 show flow-net representations of the e-mail and system log files' examples.

Not only e-mail server log files but also flow-net formats contain minimally required information, such as, where e-mails come from and are sent to, their subjects, and the date

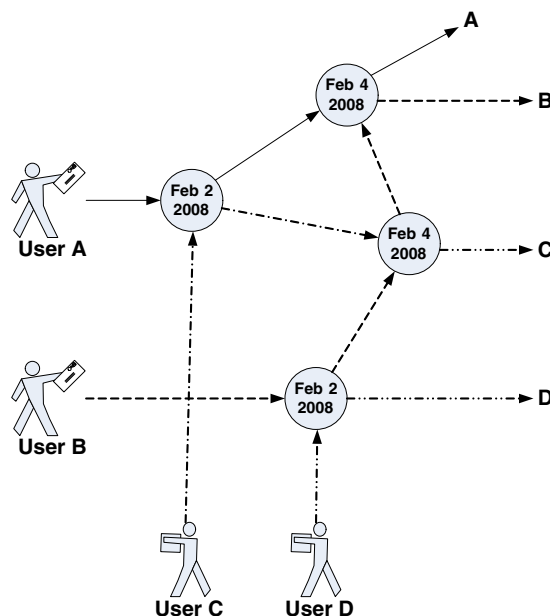


Figure 4. Examples of an original e-mail log file in an e-mail server and the resulting flow-net representation.

they were sent. Following the linked list for user A in Figure 4, we know user A sent an e-mail to user C on 2 February 2008 and to user B on 4 February 2008. Thus, in creating a URG (social network diagram), users A and B are bi-directionally connected, as shown in Figure 6. Likewise, by following user B's linked list, we see that user B sent an e-mail to user D on 2 February 2008 and then to user C on 4 February 2008, and after that he or she received another e-mail from user A. These three users are definitely connected in a URG, as shown in Figure 6. Basically, a URG is merely a social network used to complete a short period flow-net. On the other hand, URGs are constructed from e-mail or system logs over long periods that we are not concerned with. However, this weak information can be occasionally useful when information from the previous flow-nets is not sufficient. To avoid confusion, as another example, in papers [2,82], we considered covert channels of secured data in the multiuser database. However, there may be cases in which algorithms cannot generate any results, that is, any paths from one event to the target event, but we know that secrets were definitely known by users with lower priorities. In that case, we utilize URGs as auxiliary information to compensate for the lack of evidence.

Algorithm 1. Suppose that we have a flow-net data structure having information of sending-events in an e-mail server, as shown in Figure 4. Assume that we have N users. This algorithm is summarized as follows: We prepared a two-dimensional array for creating a graph data structure, where one dimension is used for senders and the other is used for recipients. In the case of N users, we prepare an $N \times N$ two-dimensional array. Each cell comprises a user ID and an integer value, which represents how many times users of the intersection interacted with each other (e.g., via e-mails, etc.).

- (1) For an $N \times N$ two-dimensional array, initialize every column with a user ID or e-mail address, and set the counter value to 0.
- (2) For each user entity (linked list), investigate events from the first to the last. When this user sent an e-mail to other users in the event, increment the counter of the corresponding user by 1. Repeat the same operation toward events that happened within a 1-year period, count them, and put the number of the e-mails on both intersection cells of a URG.

Here are some functions used in the pseudo code for the construction of a URG:

- **size[*email_flow-net*]** returns the number of entities or users in *e-mail_flow-net*.
- **email_flow-net[i]** returns an entity at position i of *e-mail_flow-net* (i.e., the linked list of user i 's events).
- **email_flow-net[i].begin** points to the first event of user i 's linked list.
- **email_flow-net[i].next** points to the next (newer) event in user i 's linked list.

Pseudo Code: URG is a graph abstract data structure represented as a two-dimensional array comprising *size*

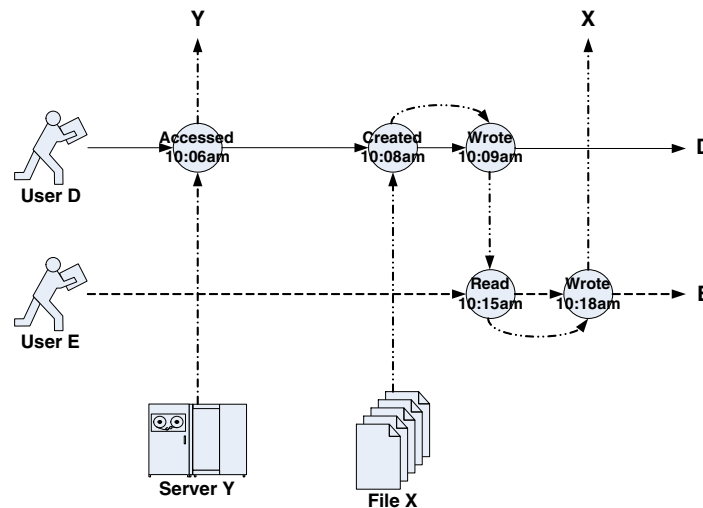


Figure 5. Examples of an original system log file and the resulting flow-net representation.

[*email_flow-net*] nodes, but initially, no connections are made among the nodes (i.e., every cell of the two-dimensional array is initialized with 0). Table I shows the log entries of the e-mails.

```
event_ptr is a pointer of an event.
email_flow-net ← web-based e-mail logs
for i = 1 to size[email_flow-net]
do event_ptr ← email_flow-net[i].begin
while event_ptr ≠ null
do j ← event_ptr → recipient
URG[i, j] = URG[i, j] + 1
event_ptr ← email_flow-net[i].next
```

Because the relationships constructed from e-mail logs are direct—that is, usually, e-mails are directly exchanged with each other, person-by-person—we call this URG a direct URG.

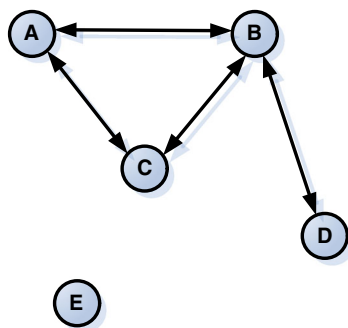


Figure 6. A user-relationship graph derived from Figure 4. From Figure 4, because user A sent e-mails to both users B and C, we suppose that they have a relationship with each other and connect them with bi-directional arrow edges. Also, user B sent both users C and D e-mails, so we connected them to represent relations. On the other hand, no one is connected to user E because user E did not interact with anyone before.

5.1.2. Step 2: creating an extended user-relationship graph.

Likewise, we can construct another URG from system log files. From the flow-net representation of a system log file in Figure 5, as we follow the links of user D's linked list, we encounter three events where user D accessed server Y and created and wrote data on file X. Therefore, when we are confronted with an event where user D accessed server Y at 10:06 AM, we next trace the links of server Y's linked list to check whether other users accessed server Y within a predefined period, and if someone did, user D is connected to those users through server Y. For example, in Figure 5, user D created file X at 10:08 AM, and following the links of file X's linked list consequently reaches an event where user E read the contents of file X at 10:15 AM. Hence, we connected users D and E in another URG (social network), as shown in Figure 7. Table II shows the related events.

Algorithm 2. Suppose that we have a flow-net data structure having the information of only indirect relations as shown earlier. Assume that we have N users and M resources. Assume that we are only concerned with the first indirect users. We prepare a two-dimensional array for creating a graph data structure, where one dimension is used for senders and the other is used for recipients. In the case of N users, we prepare an $N \times N$ two-dimensional array. Each cell has an integer value, which represents how many times users of the intersection interacted with each other (e.g., via e-mails, etc.).

Table I. Log entries of e-mails.

From	To	Subject	Date
User B	User D	This Weekend	2 February 2008
User A	User C	RE: Hi	4 February 2008
User B	User C	FW: Pictures	4 February 2008
User A	User B	Hello	4 February 2008

- (1) Investigate what resources user A accessed within a 1-year period. For example, we may know from a flow-net that user A accessed resources having all the odd numbers (i.e., 1, 3, 5, 7, 9).
- (2) For each resource, investigate which users accessed the resource by tracing its resource linked list, count how many times that they accessed it, and put the number of accesses on a URG. For example, from step 1 of this example, we know user A accessed resource 1 within this period, so we next investigate which user accessed resource 1 within this period. Keep investigating the resource linked list until its events are finally exhausted.
- (3) Do the same investigation as step 2 for other resources except for resources that have already been investigated.
- (4) Do the same investigation for the other users.

Here are some functions used in the pseudocode for the construction of an EURG:

- **size[system_flow-net]** returns the number of entities or users in *system_flow-net*.
- **system_flow-net[i]** returns an entity at position *i* of *system_flow-net*, that is, the linked list of user *i*'s events.
- **system_flow-net[i].begin** points to the first event of user *i*'s linked list.
- **event.next_subject** moves events in a user's linked list one event next (newer) from the current event and returns the pointer to a new event.
- **event.next_object** moves events in a resource's linked list one event next (newer) from the current event and returns the pointer to a new event.

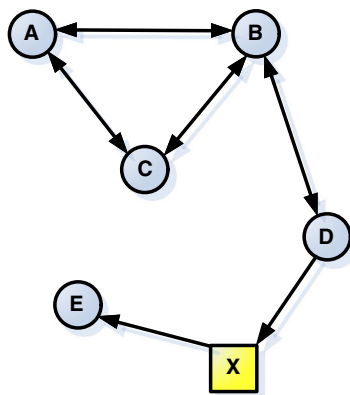


Figure 7. Extended user-relationship graph derived from Figures 4 and 6. From Figure 4, we know that user E read some data of file X, which user D created and may have put some information in. Therefore, we suppose that there is some sort of relation between users D and E represented directional edges interleaved by another object, file X, which is enclosed by a yellow square.

Table II. Table of events.

Subject	Action	Object	Date
User D	Accessed	Server Y	10:06 AM
User D	Created	File X	10:08 AM
User D	Wrote	File X	10:09 AM
User E	Read	File X	10:15 AM
User E	Wrote	File X	10:18 AM

Pseudo Code: URG is a graph abstract data structure represented as a two-dimensional array comprising size [system_flow-net] nodes, but initially, no connections are made among the nodes, that is, every cell of the two-dimensional array is initialized with 0.

```

event_ptr is a pointer of an event.
system_flow-net ← Web-based e-mail logs
for i = 1 to size[system_flow-net]
do event_ptr1 ← system_flow-net[i].begin
while event_ptr1 ≠ null
do event_ptr2 ← event_ptr1 → next_object
while event_ptr2 ≠ null
do URG.add(event_ptr2 → subject)
event_ptr2 ← event_ptr2 → next_object
event_ptr1 → next_subject

```

5.2. Putting weight/probability on user-relationship graph

The weight is the number of messages communicated between two users within a period (e.g., 1 year), normalized by the total number of messages among all users. For example, user A interacted with user B via 100 sending and receiving e-mails, and a total of 10 000 e-mails were transmitted among all users in the organization (e.g., user A through user Z) within the last year. In this case, we calculate the weight or transition probability of the link between users A and B as $100/10\,000 = 0.1$.

Here is another very simple example. Table III shows e-mail transactions among five people over a very short period. From this e-mail log file in an e-mail server, we can

Table III. E-mail log file in an e-mail server, where 10 e-mails were exchanged among five people for 5 days.

No.	From	To	Subject	Date
1	User A	User C	Hello	1 February 2008
2	User B	User D	Pictures	1 February 2008
3	User C	User A	RE: Hello	1 February 2008
4	User D	User E	FW: Pictures	1 February 2008
5	User B	User A	Question?	2 February 2008
6	User E	User D	RE: FW: Pictures	2 February 2008
7	User A	User B	RE: Question?	3 February 2008
8	User A	User C	RE: RE: Hello	4 February 2008
9	User D	User E	Last Weekend	4 February 2008
10	User E	User D	This Weekend	5 February 2008

systematically construct a flow-net and a URG, which are shown in Figures 8 and 9, respectively.

6. EXPERIMENTS WITH USER-RELATIONSHIP GRAPHS

We have implemented a URG. We developed a Java graphic user interface based on JGraph [86]. The logs in the e-mail servers were used to create the URG, which shows the direct relationships between people.

6.1. Implementation

The URG concerns direct connections between people in a network. The e-mail server logs are chosen as an example through which to construct the URG because the e-mails show the direct relations between two people or among several people. We analyze the logs from the e-mail server and generate the URG in this section.

A database is used in this implementation in order to handle more data. We select the free version of MYSQL [87] as the relational database management system. Currently, there are three tables designed for the program: user, mail_record, and file_access_log. The 'user' table contains the basic user information such as user ID, user name, and so on. The 'mail_record' table includes the logs on the mail server, and the 'file_access_log' includes the file access information on the file server. The data comprised by the 'mail_record' and 'file_access_log' tables are acquired by analyzing the logs on the mail and file servers. We will skip the analysis of the logs and use the abstract data to create the tables.

The program is designed with Java [88] based on Swing and JGraph. The program may need the MYSQL-java connector [89] to connect to the MYSQL database server. The interface is shown in Figure 10.

The 'Show mail record' button will show the table of 'mail_record' in the data base, including the record ID, sender, recipient, and send time. The 'Show relation' button

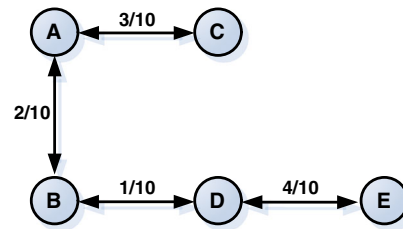


Figure 9. User-relationship graph with weights between two people. The numbers in the left side of a slash represent how many e-mails are exchanged between two people, and the ones in the right side represent the total e-mails exchanged in this e-mail server.

will display the relationships among the people in the record as in Figure 11.

As shown in Figure 11, User 'a' had sent e-mails to user 'b' and user 'c'. User 'b' had sent mail to user 'c' and user 'd'. However, there are no e-mails between user 'e' and other users.

This implementation may support large amounts of data. Because of the screen limitation, we only show five users and six mail logs in Figure 11. Still, we can select a fixed person and show all relations from other people to the person.

6.2. Evaluation

We calculate the probability of finding the information leakage when using the URG as the supplement to the flow-net. At the same time, we show the advantages of the URG.

Two matrices will be created to store the data. Either the URG or the EURG will generate an $N \times N$ two-dimensional array, with N being the number of users. The $N \times N$ two-dimensional array actually consists of an adjacency matrix, and we will use this idea of an adjacency matrix.

6.2.1. Adjacency matrix.

The adjacency matrix is similar to the URG. In a graph of $G=(V, E)$, V is the set of all vertices, and E is the set of all

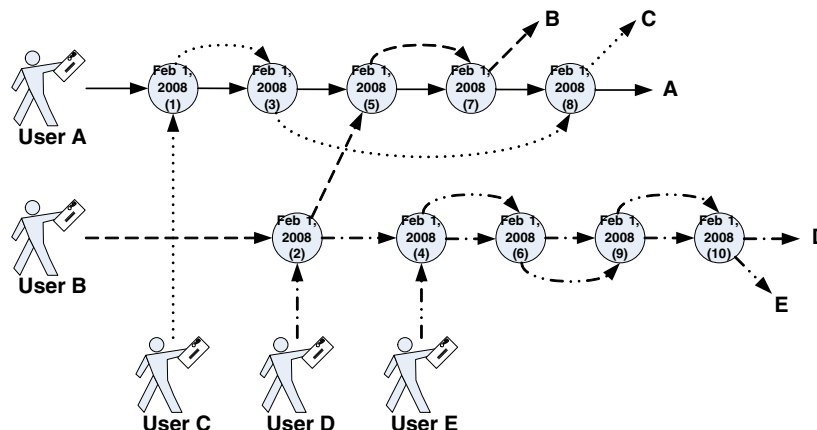
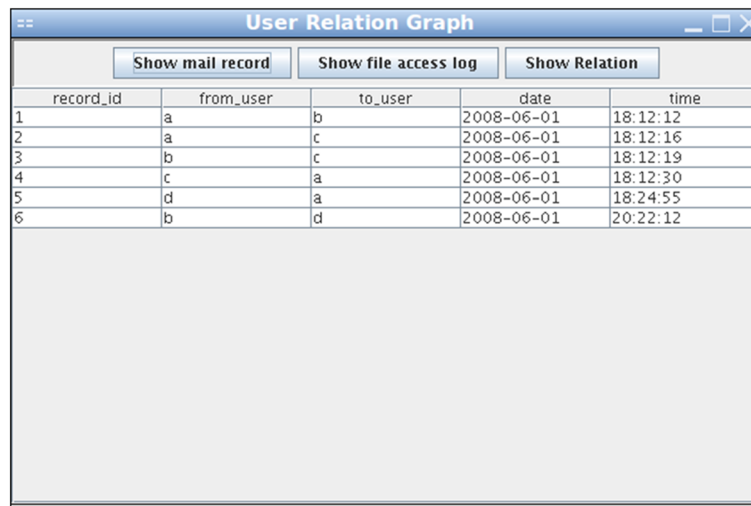


Figure 8. Flow-net representation of an auditing log file of Table III.



The interface titled "User Relation Graph" has three buttons: "Show mail record", "Show file access log", and "Show Relation". The "Show mail record" button is selected, displaying a table with the following data:

record_id	from_user	to_user	date	time
1	a	b	2008-06-01	18:12:12
2	a	c	2008-06-01	18:12:16
3	b	c	2008-06-01	18:12:19
4	c	a	2008-06-01	18:12:30
5	d	a	2008-06-01	18:24:55
6	b	d	2008-06-01	20:22:12

Figure 10. User interface.

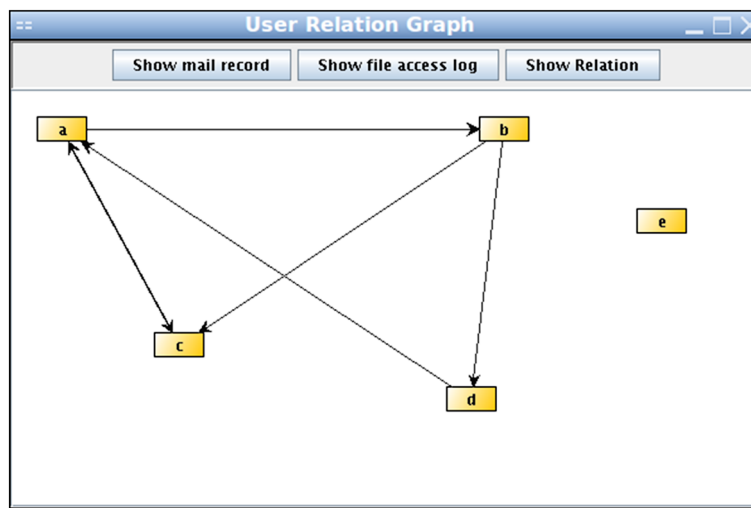


Figure 11. User relation graph.

edges existing in the graph. The users in the URG are represented by the vertices in the graph, and the relations between the users are represented by the edges in the graph. While in the adjacency matrix, the n users formed the n rows and n columns matrix, and each cell value in the matrix showed the direct relationship among the users. A value of 1 shows a direct relation between two users, and a value of 0 shows that there is no direct relation between the two users. Figure 12 is an example of the transformation from the URG to the adjacency matrix.

We call the adjacency matrix A . Assume that the variables in the X -axis and the Y -axis are i and j , respectively. Then, the value of $A[i, j]$ is set to 1 when user i has a direct relation to user j . Otherwise, the value of $A[i, j]$ is set to 0. We assume that each user is connected to himself and the value of $A[i, i]$ in the adjacency matrix is set to 1.

According to the characteristics of adjacency matrix A , we use the power n of matrix A to describe the indirect relationships among the users. The indirect relationship means that the related users have relations with each other through one or more other users. The element A_{ij}^n in the matrix A^n shows whether there are paths from vertex i to vertex j in matrix A within $n - 1$ steps. As a result, we can use this method to track the information leakage. Assume that there are m users in the relation graph. If one user cannot reach another user in $m - 1$ steps, then the two users have no relation. This means that if A_{ij}^m in the matrix A^m is 0, then user i and user j have no relation.

6.2.2. The probability of tracking the information leakage.

First, an $N \times N$ matrix A was defined to store the relations among the N users. Then, M direct relations were

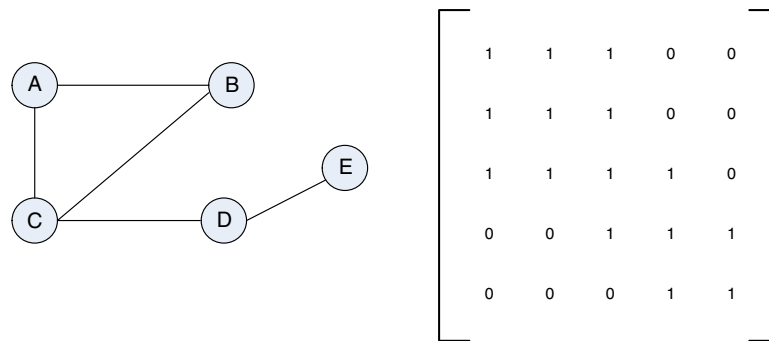


Figure 12. Transformation from the user-relationship graph to the adjacency matrix.

put into the matrix randomly. Next, we calculate the value of A^N to show the relations between each pair of users. Figure 13 shows an example of the matrices A and A^N where $N = 10$ and $M = 10$.

In the end, the probability of whether two users are related can be calculated. Figure 14 shows the relation between the number of relations put into the matrix and the probability of whether the two randomly selected users are related.

In Figure 14, we evaluate the probability of tracking the information leakage by randomly putting 1 to 30 relations to 30 and 50 users, separately. For each probability, we calculate 50 times and use the average value. The red line includes 30 users and the blue line includes 50 users. From Figure 14, we can see that the probability of tracking the information leakage will increase when we put more relations in the user relation matrix.

6.2.3. Effects of adding an extended user-relationship graph.

This section will show that the EURG may enhance the probability of tracking the information leakage. In the evaluation, we use 50 users to construct the matrix. There are 30 relationships randomly put into the 50 users. At first, we do not use the EURG, and the result is the red line in Figure 15.

The red line shows that the probability of tracking the information leakage will keep the same value (about 0.16). Then we randomly put 1 to 20 relationships in the EURG for the 50 users and add the EURG to the existing matrix. The result in blue line shows that the probability of tracking the information leakage increases from 0.16 to 0.64 with the increase of the relationship number from 1 to 20. Figure 14 shows that using an EURG may increase the probability of tracking the information leakage.

$A =$

1	1	0	0	0	0	0	0	0	1
1	1	0	1	0	0	0	0	1	0
0	0	1	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	1	0	0	1	0	0	1	0
0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	1	0	1
0	1	0	0	0	1	1	0	1	1
1	0	0	0	0	0	0	1	1	1

$B =$

1	1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1
0	0	0	0	1	0	0	0	0	0
1	1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1

Figure 13. An example of the matrix.

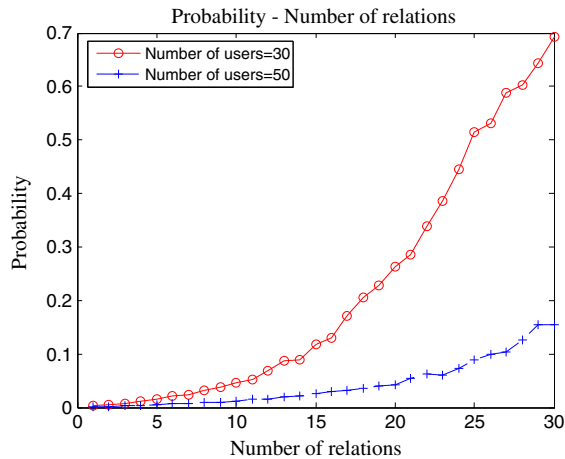


Figure 14. Probability versus the number of relations.

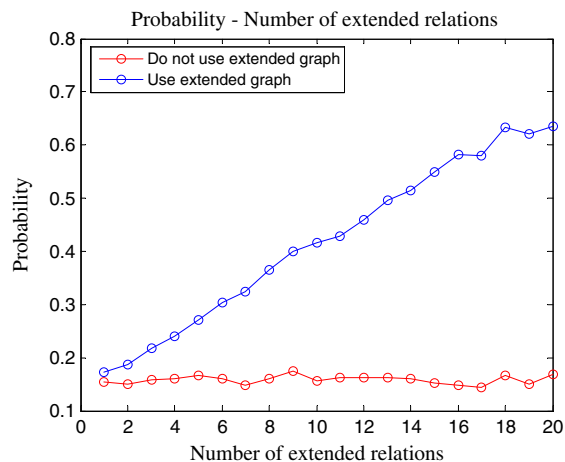


Figure 15. Probability versus the number of extended relations.

7. PROBLEMS AND CHALLENGES IN USER-RELATIONSHIP GRAPH

There are many networks other than the Internet in reality, such as telecommunications and cell phone networks. Systems include computer systems, network systems, e-mail systems, telephone systems, cellular phone systems, and so on. Typically, there exist as many networks as the number of distinct (unique) systems (organizations). The log files and flow-nets for heterogeneous systems are used separately. In other words, because each of these networks has its own communication logs, they can be used to construct flow-nets and URGs for its own purposes. For example, a telecommunication company has its own call logs, that is, who called whom, when they called, and the length of the call. From these logs, it is easy to construct flow-nets and URGs representing individuals' relations, and again, these URGs are used to discover virtually invisible links between individuals in order to compensate for the lack of information.

However, for a cluster of heterogeneous systems, aggregating every flow-net or integrating URGs into a flow-net to create a flow-net with virtual links concerning an individual has some difficulties. For example, a flow-net constructed from e-mail logs and a flow-net constructed from cell phone logs cannot have any relations to each other. More precisely, two people who are in fact identical hardly associate with each other in flow-nets in heterogeneous systems. How do we know two IDs from different heterogeneous systems come from the same person?

Moreover, we must also be concerned with privacy issues. As we mentioned before, computer forensic analysts must involve several logs in just one investigation because these log systems usually keep different aspects of criminal's activities. For example, the CARE database developed by the Care Research and Development Laboratory at the University of Alabama [90] keeps drivers' records of violations and accidents as well as the drivers' information, such as drivers' license numbers, birthdates, addresses, and so on. This information is kept at the police department and used for prediction and prevention of future accidents and violation of the traffic rules [90]. The information is also used for the early detection of criminals. The social security offices issue a social security number to each individual and keep information about the individual, which is specific to him or her, including his or her citizenship, visa status, and so on. A problem is that there are barriers that prevent the individual's privacy from penetrating through different organizations, but these barriers are necessary to maintain the confidentiality of the information. Accordingly, the situation adds difficulty to the construction of flow-nets, URGs, EURGs, and extended flow-nets. This is because the construction of flow-nets and URGs requires the cooperation of different organizations to gather other aspects of criminal behaviors.

8. CONCLUSION

Our primary concerns have been the utilization of log files of particular systems in order to support digital forensic analyses. The log files range from a simple system log file to a server side Web-based e-mail log file. Typically, log files are just users' activities ordered sequentially, and not by users or resources, but they are flawed and sometimes very confusing. In the flow-net data structure, the information of log files is optimized and organized user-by-user and resource-by-resource, and actions comprising several events (e.g., a large data transfer in the computer networks utilizes a number of packets) are pulled together into one event. In our procedure, we intend to organize all available data in log files into this beneficial flow-net data structure to allow less searching time to be required in investigations, especially in determining covert channels.

However, it is easily supposed that our investigations of covert channels are not completed because of the lack of information. In other words, although we can specify who leaks a secret and who receives it, we cannot find a possible route on which this secret passed. Thus, to enhance our investigations of covert channels, in this paper, we designed

auxiliary steps to compensate for this lack of information. Two new concepts were integrated into our flow-net data structure: a URG, or social network, and k -connectivity. In addition, according to the type of the original data, we designed two kinds of URGs, and they are called a regular URG and EURG. In regular URGs, all the connections involved in a social network are direct. For example, because Web-based e-mail log files provide direct relations of users, URGs constructed from these data are regular URGs. On the other hand, in the system log files, events do not show direct connections of users, but instead, users are connected through some resources that they share. Therefore, URGs constructed from this kind of data are indirect. Accordingly, we aggregate these two kinds of URGs into one constructing another URG, called an EURG.

Finally, we put weights of links of the graphs to show the likelihood of a user knowing another, so our investigations rely on these probabilities to construct virtual links in constructed flow-net data structures.

Experiments show that the use of EURGs may increase the probability of tracing the information leakage.

ACKNOWLEDGEMENTS

This work is supported in part by the US National Science Foundation (NSF) under the grant numbers CNS-0737325, CNS-0716211, CCF-0829827, and CNS-1059265.

REFERENCES

1. Kruse WG II, Heiser JG. *Computer Forensics: Incident Response Essentials*. Addison-Wesley Professional, 2001.
2. Takahashi D, Xiao Y. *Retrieving knowledge from auditing log files for computer and network forensics and accountability*, Vol. 1, (Wiley) Security and Communication Networks. No. 2, Feb. 29, 2008; 147–160.
3. Adamic L, Adar E. How to search a social network. *Social Networks* 2005; **27**(3): 187–203.
4. Killworth P, Bernard H. Reverse small world experiment. *Social Networks* 1978; **1**: 159–192.
5. Killworth PD, Bernard HR. A pseudomodel of the small world problem. *Social Forces* 1979; **58**(2): 477–505.
6. Lunberg CC. Patterns of acquaintanceship in society and complex organization: a comparative study of the small world problem. *The Pacific Sociological Review* 1975; **18**: 206–222.
7. Milgram S. The small-world problem. *Psychology Today* 1967; **1**: 62–67.
8. Travers J, Milgram S. An experimental study of the small world problem. *Sociometry* 1969; **32**: 425–443.
9. Dodds PS, Muhamad R, Watts DJ. An experimental study of search in global social networks. *Science* 2003; **301**: 827–829.
10. Watts DJ, Dodds PS, Newman MEJ. Identity and search in social networks. *Science* 2002; **296**: 1302–1305.
11. Kleinberg J. Navigation in a small world. *Nature* 2000; **406**.
12. Kleinberg J. Small-world phenomena and the dynamics of information. *Advances in Neural Information Processing Systems (NIPS)* 2001; **14**.
13. Granovetter S. The strength of weak ties. *The American Journal of Sociology* 1973; **78**: 1360–1380.
14. Khuller S, Raghavachari B. Improved approximation algorithms for uniform connectivity problems. *The 27th Annual ACM Symposium on Theory of Computing (STOC)*, 1995.
15. Fu B, Xiao Y. An implementation scheme of flow-net and its applications on detecting attacks in wireless networks. *Proc. of IEEE GLOBECOM* 2010.
16. Xiao Y, Meng K, Takahashi D. Accountability using flow-net: design, implementation, and performance evaluation. *(Wiley Journal of) Security and Communication Networks*, Special Issue on Security and Privacy in Emerging Information Technologies, accepted, DOI: 10.1002/sec.348.
17. Takahashi D, Xiao Y, Meng K. Creating user-relationship-graph in use of flow-net and log files for computer and network accountability and forensics. *Proceedings of the IEEE Military Communications Conference* 2010 (IEEE MILCOM 2010).
18. Xiao Y, Yue S, Fu B, Ozdemir S. GlobalView: building global view with log files in a distributed/networked system for accountability. *(Wiley Journal of) Security and Communication Networks*, accepted, DOI: 10.1002/sec.374.
19. Liu J, Xiao Y. Temporal accountability and anonymity in medical sensor networks. *ACM/Springer Mobile Networks and Applications (MONET)*, Special issue on Ubiquitous Body Sensor Networks, accepted, DOI: 10.1007/S11030-010-0254-6.
20. Takahashi D, Xiao Y, Zhang Y, Chatzimisios P, Chen HH. IEEE 802.11 user fingerprinting and its applications. *Computers and Mathematics with Applications* 2010; **60**(2): 307–318.
21. Meng K, Xiao Y, Vrbsky SV. Building a wireless capturing tool for WiFi. *(Wiley Journal of) Security and Communication Networks* 2009; **2**(6): 654–668.
22. Y Xiao. Accountability for wireless LANs, ad hoc networks, and wireless mesh networks. *IEEE Communication Magazine*, Special Issue on Security in Mobile Ad Hoc and Sensor Networks. 2008; **46**(4): 116–126. DOI: 10.1109/MCOM.2008.4481350.
23. Takahashi D, Xiao Y. Retrieving knowledge from auditing log files for computer and network forensics and accountability. *(Wiley Journal) Security and Communication Networks* 2008; **1**(2): 147–160, DOI: 10.1002/sec.10.

24. Fu B, Xiao Y. Q-accountable: a overhead-based quantifiable accountability in wireless networks. Proceedings of IEEE Consumer Communications and Networking Conference (IEEE CCNC 2012).
25. Xiao Z, Y Xiao, Du D. Building accountable smart grids in neighborhood area networks. Proceeding of The IEEE Global Telecommunications Conference 2011 (IEEE GLOBECOM 2011).
26. Zeng L, Chen H, Xiao Y. Accountable administration and implementation in operating systems. Proceeding of The IEEE Global Telecommunications Conference 2011 (IEEE GLOBECOM 2011).
27. Xiao Z, Xiao Y. Accountable MapReduce in cloud computing. Proceedings of The IEEE International Workshop on Security in Computers, Networking and Communications (SCNC 2011).
28. Liu J, Xiao Y, Gao J. Accountability in smart grids. Proceedings of IEEE Consumer Communications and Networking Conference 2011 (IEEE CCNC 2011), Smart Grids Special Session.
29. Xiao Z, Xiao Y, Wu J. A quantitative study of accountability in wireless multi-hop networks. Proceedings of 2010 39th International Conference on Parallel Processing (ICPP 2010).
30. Xiao Z, Xiao Y. PeerReview analysis and re-evaluation for accountability in distributed systems or networks. Proceedings of the 4th International Conference on Information Security and Assurance (ISA2010), CCIS 76, 149–162, 2010.
31. Xiao Z, Xiao Y. P-accountable networked systems. Proceeding of INFOCOM 2010, Work in Progress (WIP) Track, acceptance rate is 28% (27 over 97).
32. Ramsey BW, Mullins BE, Thomas RW, Andel TR. Subjective audio quality over a secure IEEE 802.11n network. *International Journal of Security and Networks* 2011; **6**(1): 53–63.
33. Xiao Y. Editorial. *International Journal of Security and Networks* 2011; **6**(1): 1–1.
34. Kundur D, Feng X, Mashayekh S, Liu S, Zourntos T, Butler-Purry KL. Towards modelling the impact of cyber attacks on a smart grid. *International Journal of Security and Networks* 2011; **6**(1): 2–13.
35. Kalogridis G, Denic SZ, Lewis T, Cepeda R. Privacy protection system and metrics for hiding electrical events. *International Journal of Security and Networks* 2011; **6**(1): 14–27.
36. Li F, Luo B, Liu P. Secure and privacy-preserving information aggregation for smart grids. *International Journal of Security and Networks* 2011; **6**(1): 28–39.
37. Zhang J, Gunter CA. Application-aware secure multicast for power grid communications. *International Journal of Security and Networks* 2011; **6**(1): 40–52.
38. Choi T, Acharya HB. Is that you? Authentication in a network without identities. *International Journal of Security and Networks* 2011; **6**(4).
39. Chai Q, Gong G. On the (in)security of two joint encryption and error correction schemes. *International Journal of Security and Networks* 2011; **6**(4).
40. Tang S, Li W. An epidemic model with adaptive virus spread control for wireless sensor networks. *International Journal of Security and Networks* 2011; **6**(4).
41. Luo G, Subbalakshmi KP. KL-sense secure image steganography. *International Journal of Security and Networks* 2011; **6**(4).
42. Chang W, Wu J, Tan CC. Friendship-based location privacy in mobile social networks. *International Journal of Security and Networks* 2011; **6**(4).
43. Zhao X, Li L, Xue G. Authenticating strangers in online social networks. *International Journal of Security and Networks* 2011; **6**(4).
44. Walker D, Latifi S. Partial iris recognition as a viable biometric scheme. *International Journal of Security and Networks* 2011; **6**(2/3).
45. Desoky A. Edustega: an education-centric steganography methodology. *International Journal of Security and Networks* 2011; **6**(2/3).
46. Ampah N, Akujuobi C, Alam S, Sadiku M. An intrusion detection technique based on continuous binary communication channels. *International Journal of Security and Networks* 2011; **6**(2/3).
47. Chen H, Sun B. Editorial. *International Journal of Security and Networks* 2011; **6**(2/3).
48. Barua M, Liang X, Lu R, Shen X. ESPAC: enabling security and patient-centric access control for eHealth in cloud computing. *International Journal of Security and Networks* 2011; **6**(2/3).
49. Jaggi N, Reddy UM, Bagai R. A three dimensional sender anonymity metric. *International Journal of Security and Networks* 2011; **6**(2/3).
50. Sharma MJ, Leung VCM. Improved IP multimedia subsystem authentication mechanism for 3G-WLAN networks. *International Journal of Security and Networks* 2011; **6**(2/3).
51. Cheng N, Govindan K, Mohapatra P. Rendezvous based trust propagation to enhance distributed network security. *International Journal of Security and Networks* 2011; **6**(2/3).
52. Fathy A, ElBatt T, Youssef M. A source authentication scheme using network coding. *International Journal of Security and Networks* 2011; **6**(2/3).
53. Liu L, Xiao Y, Zhang V, Faulkner A, Weber K. Hidden information in Microsoft Word. *International Journal of Security and Networks* 2011; **6**(2/3).

54. Chow SSM, Yiu S. Exclusion–intersection encryption. *International Journal of Security and Networks* 2011; **6**(2/3).
55. Chinnappen-Rimer S, Hancke GP. Actor coordination using info-gap decision theory in wireless sensor and actor networks. *International Journal of Sensor Networks* 2011; **10**(4): 177–191.
56. Zhang F, Dojen R, Coffey T. Comparative performance and energy consumption analysis of different AES implementations on a wireless sensor network node. *International Journal of Sensor Networks* 2011; **10**(4): 192–201.
57. Xiao Y, Takahashi D, Liu J, Deng H, Zhang J. Wireless telemedicine and m-health: technologies, applications and research issues. *International Journal of Sensor Networks* 2011; **10**(4): 202–236.
58. Rosi A, Berti M, Bicocchi N, Castelli G, Corsini A, Mamei M, *et al.* Landslide monitoring with sensor networks: experiences and lessons learnt from a real-world deployment. *International Journal of Sensor Networks* 2011; **10**(3): 111–122.
59. Wang F, Zeng P, Yu H, Xiao Y. Error compensation algorithm in wireless sensor networks synchronisation. *International Journal of Sensor Networks* 2011; **10**(3): 123–131.
60. Toscani D, Giordani I, Cislighi M, Quarenghi L. Querying sensor data for environmental monitoring. *International Journal of Sensor Networks* 2011; **10**(3): 132–142.
61. Fedor S, Collier M, Sreenan CJ. Cross-layer routing and time synchronisation in wireless sensor networks. *International Journal of Sensor Networks* 2011; **10**(3): 143–159.
62. Mishra V, Mathew J, Pradhan DK. Fault-tolerant de-Bruijn graph based multipurpose architecture and routing protocol for wireless sensor networks. *International Journal of Sensor Networks* 2011; **10**(3): 160–175.
63. Tseng C, Lee S, Prasad NR, Wódczak M. Editorial. *International Journal of Sensor Networks* 2011; **10**(1/2): 1–2.
64. Kafetzoglou S, Papavassiliou S. Energy-efficient framework for data gathering in wireless sensor networks via the combination of sleeping MAC and data aggregation strategies. *International Journal of Sensor Networks* 2011; **10**(1/2): 3–13.
65. Krontiris I, Dimitriou T. Scatter—secure code authentication for efficient reprogramming in wireless sensor networks. *International Journal of Sensor Networks* 2011; **10**(1/2): 14–24.
66. Boubiche D, Bilami A. HEEP (Hybrid Energy Efficiency Protocol) based on chain clustering. *International Journal of Sensor Networks* 2011; **10**(1/2): 25–35.
67. Soliman H, Al-Otaibi M. Enhancing AODV routing protocol over mobile ad hoc sensor networks. *International Journal of Sensor Networks* 2011; **10**(1/2): 36–41.
68. Jaggi N, Kar K. Multi-sensor activation for temporally correlated event monitoring with renewable energy sources. *International Journal of Sensor Networks* 2011; **10**(1/2): 42–58.
69. Lu L, Wu JC, Chen S. A cluster-based algorithm for redundant nodes discovery in dense sensor networks. *International Journal of Sensor Networks* 2011; **10**(1/2): 59–72.
70. Morreale P, Qi F, Croft P. A green wireless sensor network for environmental monitoring and risk identification. *International Journal of Sensor Networks* 2011; **10**(1/2): 73–82.
71. Wódczak M. Autonomic cooperative networking for wireless green sensor systems. *International Journal of Sensor Networks* 2011; **10**(1/2): 83–93.
72. Poornima AS, Amberker BB. PERSEN: power-efficient logical ring based key management for clustered sensor networks. *International Journal of Sensor Networks* 2011; **10**(1/2): 94–103.
73. Li F. Adaptive resource allocation in multiuser cooperative networks with proportional rate constraints. *International Journal of Sensor Networks* 2011; **10**(1/2): 104–110.
74. Amundson I, Sallai J, Koutsoukos X, Ledeczi A, Maroti M. RF angle of arrival-based node localisation. *International Journal of Sensor Networks* 2011; **9**(3/4): 209–224.
75. Shen C, Hong Y-WP, Chao C, Yang S. Editorial. *International Journal of Sensor Networks* 2011; **9**(3/4): 121–123.
76. Arienzo L, Longo M. Energy-efficient collaborative tracking in wireless sensor networks. *International Journal of Sensor Networks* 2011; **9**(3/4): 124–138.
77. Azad AP, Chockalingam A. Enhancing lifetime of wireless sensor networks using multiple data sinks. *International Journal of Sensor Networks* 2011; **9**(3/4): 139–157.
78. Majumdar A, Ward RK. Increasing energy efficiency in sensor networks: blue noise sampling and non-convex matrix completion. *International Journal of Sensor Networks* 2011; **9**(3/4): 158–169.
79. Jedermann R, Becker M, Gorg C, Lang W. Testing network protocols and signal attenuation in packed food transports. *International Journal of Sensor Networks* 2011; **9**(3/4): 170–181.
80. Taniguchi Y, Kitani T, Leibnitz K. A uniform airdrop deployment method for large-scale wireless sensor networks. *International Journal of Sensor Networks* 2011; **9**(3/4): 182–191.
81. Tran TD, Agbinya JI, Al-Jumaily AA. Per node deployment based detection of controlled link establishment

- attack in distributed sensor networks. *International Journal of Sensor Networks* 2011; **9**(3/4): 192–208.
82. Takahashi D, Xiao Y. Complexity analysis of retrieving knowledge from auditing log files for computer and network forensics and accountability. *Proc. of IEEE ICC* 2008, 1474–1478.
83. Xiao Y. Flow-net methodology for accountability in wireless networks. *IEEE Network*, Vol. 23, No. 5, Sept./Oct. 2009, 30–37.
84. Xiao Y, Meng K, Takahashi D. Implementation and evaluation of accountability using flow-net in wireless networks. *Proc. of IEEE MILCOM* 2010
85. Elmasri R, Navathe SB. *Fundamentals of Database Systems*, 5th edn, Pearson Education, Benjamin/Cummings, 2006.
86. <http://www.jgraph.com>
87. <http://www.mysql.com/>
88. <http://www.java.com>
89. <http://www.mysql.com/products/connector/j/>
90. Ding L, Dixon B. Using an edge-dual graph and k -connectivity to identify strong connections in social networks. *Proc. of the ACM 46th Annual Southeast Regional Conference*, 475–480, 2008.