Computer Network Intrusion Detection, Assessment And Prevention Based on Security Dependency Relation

Stephen S. Yau and Xinyu Zhang
Computer Science and Engineering Department
Arizona State University
Tempe, AZ 85287-5406, U.S.A.
{yau, zhxinyu}@asu.edu

Abstract

In this paper an approach to detection, assessment and prevention of further intrusions of distributed intrusions in a computer network is presented. Our approach uses audit data from multiple network nodes and services. To achieve accurate result, inherent security relations among different network nodes should be considered. In our approach, security dependency relation (SDR) is defined to describe these relations, and ripple effect analysis is used to detect, assess, and prevent intrusions based on SDRs. Agents are used to improve the scalability and efficiency of our approach.

Keywords: *Network security, intrusion detection, intrusion assessment, intrusion prevention, ripple effect analysis, security dependency relation, agent.*

1 Introduction

In order to make a computer network secure and robust, effective methods are needed for intrusion detection, assessment, and prevention in the network. There have been many intrusion detection systems (IDSs) [1], most of which are host-based and do not cover distributed intrusions involving several network nodes. There are two kinds of approaches to network intrusion detection: network-based intrusion detection (NID) and distributed intrusion detection (DID). A NID approach [1] monitors network communication to detect attacks to a network, but it cannot detect operating system level intrusions and frequently cannot handle data in a large network area. A DID approach [2][3] makes use of data from multiple nodes and services, and thus has more chances than other approaches to detect distributed intrusions. Our approach belongs to DID and can be applied to an enterprise-wide computer network which has centralized control [4]. In this paper, the computer networks are enterprise-wide computer networks.

DID across a network will impose the following requirements to IDS:

- IDS needs to collect selected data from different sites.
- IDS should have an effective way to analyze audit data generated by different network nodes for detecting intrusions.
- IDS should have good scalability.

Currently, very little work has been done to address intrusion assessment [5] which is very important to minimize the impact on network security when an intrusion has been detected. Compared to intrusions which involve only one machine, distributed intrusions which occur across a network are more complicated and time-consuming to assess, and thus automated efficient intrusion assessment is needed.

It is very difficult to prevent intrusions from occurrence. However, since distributed intrusions across a network will take a period of time to spread from node to node, by adjusting the system configuration, this kind of intrusions can be prevented by stopping them from reaching the goal of breaking the most important part of the network.

In this paper, we will present an approach to detection, assessment, and prevention of further intrusions of distributed intrusion in a computer network. Inherent security relations, which can affect intrusion propagation, will be described by security dependency relation (SDR). Based on SDR, ripple effect analysis is used to collect necessary data from different nodes. The collected data is then analyzed to detect intrusions which may have occurred across a network. At the same time, the intrusion scope across a network is automatically assessed and response is automatically made to stop the intrusion to facilitate further intrusions to other nodes.

In our approach, we will use agents [6] to improve the scalability and efficiency of our approach.

0-7695-0368-3/99 $10.00 © 1999 IEEE

2 Security Dependency Relation and Intrusion Ripple Effect

Since our approach is based on SDRs and ripple effect analysis of intrusion, we will discuss these in this section first.

When an intrusion occurs at a node, the nature of SDR makes the intrusion very likely propagates along SDRs to SDR-related nodes. Our approach is based on the fact that the anomaly caused by an intrusion will propagate along SDRs, and hence at the SDR-related nodes anomaly will likely be found.

A typical intrusion across network consists of several steps [7] and a sub-goal is accomplished at each step. Each sub-goal is to help the intruder accomplish the next sub-goal. Finally, the most critical part is broken. Typically, the first step is that an account is intruded using "password cracker" program, or simply stealing the password, or through the security holes. Then using Trojan Horses or other system facilities, more and more nodes are intruded. Our approach is to prevent intrusions of this pattern.

A SDR is said to exist from node A to node B in a network if some programs or services in node A can provide significant help to an intruder to break node B after the intruder has broken node A. Two nodes are said SDR-related if they have SDRs between them. The effectiveness of our approach depends on how accurate and complete we can identify SDRs among all the nodes. SDRs are constructed beforehand.

A SDR_{AB} from node A to node B is a 4-tuple: (A, S_A, B, S_B), where S_A is a subject at A, S_B is a subject at node B. A subject may be a user account, a service, or an application, which needs privilege to access. There may be more than one SDR between two nodes.

According to the construction process, SDRs can be categorized into two kinds:

- **Automatically constructed** Some SDRs can be constructed automatically from the system configuration information. For example, the most common SDR is access relation, including access to user account, network resource, database, etc.
- **Manually constructed** Some SDRs come from distributed applications running on computer network. These applications will transport some information which cannot be controlled by a network system, but will harm network system security. We need to understand these applications in detail to identify SDRs.

SDRs are transitive. If A and B, B and C have SDRs, and if A is intruded, B may also be intruded because of the SDR_{AB} and C may also be intruded because of the possible intrusion of B and SDR_{BC}.

3 Our Approach

Intrusion detection approaches can be classified in two categories: profile-based and signature-based. A profile-based approach detects anomalies by comparing the system behavior with a profile which is used to describe the normal system behavior. A signature-based approach searches for known intrusion patterns to detect misuse of a network. Our approach is profile-based and detects only those intrusions producing identifiable anomalies.

Based on component-based software development [8], our approach consists of primarily three components: Intrusion Detection Unit (IDU), Analysis Component (AC), and Reconfiguration Component (RC). Two additional modules are used for system maintenance: Profile Management module, and SDR Management module. The system architecture is shown in Figure 1.

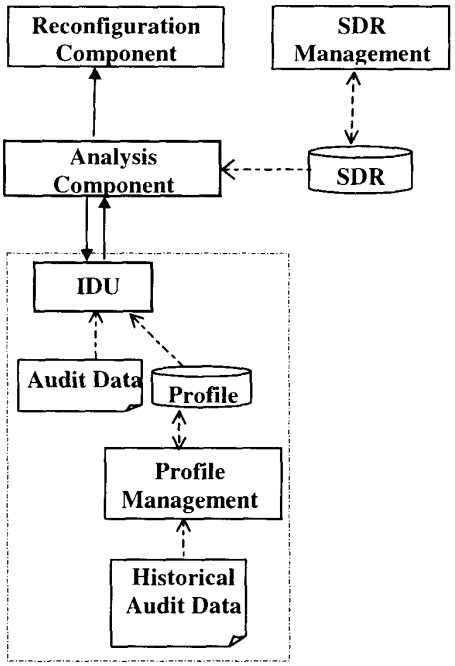

Figure 1 The system architecture of our approach

An IDU is an autonomous agent [6] which performs local intrusion detection using a profile-based intrusion detection system IDES [9], which describes the normal system behavior and detect anomalies by calculating the degree of deviation of current system behavior from the normal system

behavior. IDU scans audit data once every short period, such as 5 minutes, and calculate the degree of deviation for each subject. In our approach, we normalize the degree of deviation as the probability of intrusion (POI) of a subject, which denotes how probable an intrusion may have occurred to the subject of a node.

An IDU has the following functions:

- Performing local intrusion detection.
- Invoking an AC which performs data collection and analysis.
- Accepting request from an AC to detect anomaly of a subject and sending results back to the AC.

IDUs can be configured at run time.

Using IDUs, local audit data is processed by the local IDU. Only global data, such as the intrusion detection result, is transported across the network. This feature improves the efficiency of our approach.

An Analysis Component (AC) is an agent for collecting and analyzing data. It can collect results generated by IDUs at different nodes, and analyze the collected data to calculate the intrusion detection result. If some abnormal behavior is detected at one node, an AC will be informed to find SDR-related nodes and IDUs at these nodes will be invoked to check the system status and send the result to the AC. The collected data is analyzed to confirm whether intrusions have occurred. At the same time, the intrusions are assessed. Then the AC inform RCs to prevent the intrusions.

ACs are organized in a hierarchical structure, as shown in Figure 2. They will collaborate with each other to invoke IDUs in different nodes and collect data from these IDUs. This makes our approach scalable in a large network. Each AC in the lowest level connects to several IDUs. Each IDU is connected to only one AC.

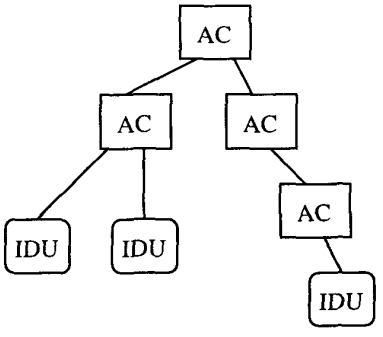

Figure 2 The hierarchical Structure of the Analysis Component

An RC is an agent for initiating actions to isolate the intrusions at a node. There is one RC at each machine. With the help of the operating system and applications which has registered in an RC, the RC can do the following:

- Changing the configuration of applications and services, such as changing the security level of services.
- Closing user accounts.
- Invoking and stopping applications.
- Moving the applications and services to other nodes by communicating with RCs at these nodes.

Before our approach starts to be applied to a network, we need to construct profiles of subjects. For each subject, we use several measures to check the subject behavior, such as the CPU utilization time of the subject per day, the number of authentication failures of the subject per day [9][10]. We need two threshold values to detect intrusions: lower-boundary and upper-boundary. Since it is difficult to know the actual number of intrusions to the network from historical audit data, we may calculate the approximate threshold values as follows:

(1) Use the most recent audit data, once every short period, to calculate the POI value for each subject.

(2) Calculate the two threshold values as follows:

lower-boundary = average(POIs),

upper-boundary = [min(POIs), max(POIs)]/2,

where POIs are all POI values of all subjects we got from the historical audit data.

In a network with security protection, there will be only a few anomalies. Since the POIs of normal behavior are very small, most POIs will be very small and a few will be very large. This will make the lower-boundary greater than the POIs of normal behavior and smaller than the upper-boundary.

Because profile-based intrusion detection approach has the disadvantage that it is difficult to find proper thresholds, we will adjust these two threshold values based on the intrusion detection as follows: If we have many false alarms, we can reduce the false alarms by increasing these two threshold values a little for the following reasons: Anomalies with POIs greater than the upper-boundary will be considered as intrusions, and the increase of upper-boundary will decrease the number of POIs greater than it. Anomalies with POIs greater than the lower-boundary will be investigated, and the increase of lower-boundary will decrease the number of POIs greater than it. Similarly, if we miss many intrusions, we will decrease these two threshold values a little.

Our approach can be summarized in the following steps, and these steps will be elaborated further in subsequent subsections.

Step 1. Monitoring the system status.

IDUs are running in different nodes and calculate the POIs of subjects once every short period using the process to be described in Section 3.1. A variable *"intruded"* with value *"yes"* or *"undetermined"* is used to record the intrusion detection status.

If the POI of a subject at node NODE is found greater than or equal to the upper-boundary, an intrusion is detected, and the variable *"intruded"* is assigned the value *"yes"*, and go to Step 2. If the POI of a subject SUB at a node NODE is greater than or equal to the lower-boundary but smaller than the upper-boundary, the variable *"intruded"* is assigned the value *"undetermined"*, and go to Step 2. If all POIs of subjects are smaller than the lower-boundary, continue to execute Step 1.

Step 2. Ripple effect analysis and intrusion assessment.

The AC which connects to the IDU at node NODE is provided the POI of subject SUB and the variable *"intruded"* by the IDU at node NODE. Then the AC will find all affected subjects and collect the POIs of these subjects using the ripple effect analysis algorithm to be presented in Section 3.2. At this step the intrusion assessment is performed automatically by finding all affected subjects.

If the variable *"intruded"* in Step 1 has the value *"yes"*, go to Step 4. Otherwise, go to Step 3.

Step 3. Data analysis and intrusion detection

The collected data is analyzed by the AC to calculate a POI of the network using the algorithm to be presented in Section 3.3. The POI of the network denotes the probability that an intrusion exists in the network.

If the POI of the network is greater than or equal to the upper-boundary, we decide that an intrusion has occurred, and go to Step 4. Otherwise, go back to Step 1.

Step 4. Prevention of further intrusion.

The AC sends the result of intrusion detection and assessment to RCs at intruded nodes. Then these RCs will close intruded subjects, move applications from intruded nodes to secured nodes, and raise the security level of services and applications at intruded nodes. System administrators are informed to take further actions. Finally go back to Step 1.

3.1 Monitoring the System Status

In Step 1, we monitor the system activities with IDUs.

IDUs first run at some or all nodes to monitor system activities to detect abnormal system behavior. It is ideal to have at least one IDU running at each node to monitor all activities. However, to maintain good system performance, there may be only several important nodes having IDUs running all the time, and IDUs at other nodes will be invoked when needed. At certain node, only selected activities are monitored all the time and other activities will be checked when needed.

Once every short period, an IDU uses the following steps to calculate the POI of a subject [9]:

- Calculate each measure value.
- Calculate the deviation of each measure value from the measure distribution.
- Combine deviations of all measures to one deviation value D.
- Calculate the POI value:

  $POI = \min(1, D/D_{Max})$,

  where D_{Max} is the maximum deviation value of this subject calculated from the historical audit data.

3.2 Ripple Effect Analysis and Intrusion Assessment

In Step 2, an AC will collect data from SDR-related nodes using ripple effect analysis.

A set Q is used to store all subjects from which SDRs will be checked, and a set AF is used to store all affected subjects found. We use the following ripple effect analysis algorithm to find all affected subjects and collect their POIs:

Ripple effect analysis algorithm:

Initialization: Q={Sub}, and AF={ }.

While Q is not empty do the following:

- Select one subject S_A at node A from Q
- From the SDR library, find SDRs which contain S_A and have not been checked in this process.
- If there are SDRs found:

  For each SDR found, do the following: Suppose it is in the form of (A, S_A, B, S_B) or (B, S_B, A, S_A). At node B, invoke the IDU to calculate the POI of S_B. If the POI of S_B is greater than or equal to the lower-boundary, add S_B into Q and send POI_{SB} to the AC.
- Move S_A from Q into AF.

When Q is empty, all subjects in AF are affected subjects.

If the variable *"intruded"* in Step 1 is *"yes"*, an intrusion has been detected, and hence all affected subjects are considered intruded, and all nodes which have intruded subjects are considered intruded.

3.3 Data Analysis and Intrusion Detection

In Step 3, the AC which collected data will calculate the POI of the network from the POIs of the affected subjects.

Since the POI of a subject at a node is the probability that the subject may be intruded, the probability that an intrusion exists at the network is:

$$POI_{Network} = 1 - \prod_{\text{For every affected subject S}} (1 - POI_S),$$

where POI_S denotes the POI of the subject S.

If the intrusion is detected in this step, all affected subjects of this intrusion are considered intruded, and all nodes which have intruded subjects are considered intruded.

3.4 Prevention of Further Intrusion

In order to prevent the distributed intrusions from further activities, at each intruded node, the RC is invoked by the AC performing ripple effect analysis. With the collaboration of the operating system of this node and applications registered in the AC, the RC will perform the following:

- Raising the security levels of applications and services.
- Closing the intruded accounts and stopping all applications they initiated.
- Closing intruded services and intruded applications.
- Moving applications and services which are not intruded from intruded nodes to secured nodes.

System administrators will be informed to take further actions.

4 An Example

Here we use an example to illustrate our approach.

Consider a bank system in which its machines M, A and B are behind a firewall, as shown in Figure 3. Machine M contains important information of the bank. It communicates with A and B frequently. Machines C and D are located outside the firewall. User Alice at machine C sends data to the database at machine B everyday. The database is the only subject Alice should use legally on machine B. Machine D provides NFS service to machine C, and users at C may run some application programs on D. Our IDUs are running at machines A and B all the time. An AC is running at machine B.

A malicious hacker now wants to break into the mainframe M. However, he finds that it is difficult to get through the firewall. Then somehow he steals Alice's account at C. From Alice's account he tries to access the database at machine B. During this process, he tries to break other peripheral machines, including machine D.

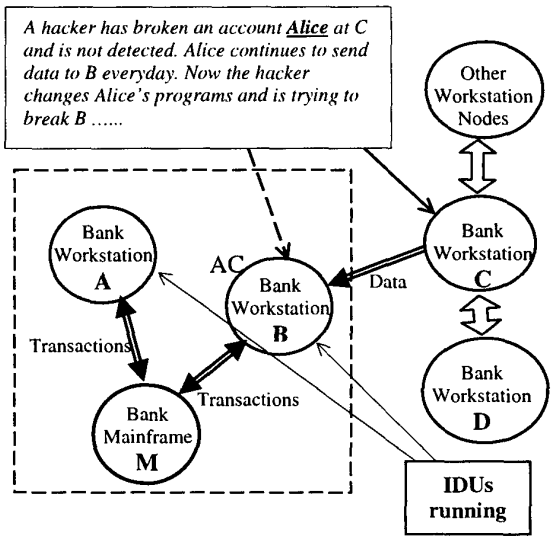

Figure 3 An example of intrusion detection, assessment, and prevention

Before our approach starts running, we scan the historical audit data to build the profiles of subjects and calculate the threshold values using the process described in Section 3. Here we have:

lower-boundary = 0.2

upper-boundary = 0.5

The system has the following SDRs:

(1). < C, Alice, B, DB-Alice >

(2). < C, Alice, D, NFS >

(3). < A, __, M, __ >

(4). < B, __, M, __ >

The following is the scenario of how the intrusion is detected, assessed, and prevented:

(1) Monitoring the system status

IDUs are running on A and B. The IDU on machine B detects anomaly of Alice and calculate the POI value 0.2. So $POI_{DB\text{-}Alice} = 0.2$. It is between the lower-boundary and upper-boundary, and hence we conclude that the variable *"intruded"* has the value *"undetermined"*.

Then the AC is informed to perform ripple effect analysis.

(2) Ripple effect analysis and intrusion assessment

Initially the set Q={DB-Alice}, and the set AF={}. Select the subject DB-Alice from Q. The AC finds that SDR(1) contains subject DB-Alice, and hence the IDU at C is invoked to check the status of

Alice on C. It finds some unexpected change at Alice's account, calculates the POI of Alice, and results in

POI_{Alice}=0.4

Add Alice into Q and move DB-Alice from Q into AF. Now Q={Alice} and AF={DB-Alice}. Select the subject Alice from Q. SDR(2) is found containing subject Alice on machine C, and hence IDU on machine D is invoked to check the status of the NFS. The IDU calculates the POI of NFS and results in

POI_{NFS}=0.2

Add NFS into Q and move Alice from Q into AF. Now Q={NFS} and AF={DB-Alice, Alice}. Next NFS is selected. No more SDRs are related with NFS. So no subject is added into Q and NFS is moved from Q inot AF.

Now Q is empty and the ripple effect analysis process ends. We have the affected subjects as follows:

AF={ DB-Alice, Alice, NFS}

Since the variable *"intruded"* is *"undetermined"*, the data analysis is performed next.

(3) Data analysis and intrusion detection

Now we have:

$POI_{DB\text{-}Alice} = 0.2$

$POI_{Alice} = 0.4$

$POI_{NFS} = 0.2$

and the POI of the network becomes

$POI_{Network} = 1-(1-0.2)(1-0.4)(1-0.2)$

$= 0.616$

Since $POI_{Network}$ is greater than the upper-boundary, the AC concludes that an intrusion has occurred.

The intrusion has already been assessed in the last step by finding all affected subjects: DB-Alice at B, Alice at C, and NFS at D. So these subjects and node B, C, D are considered intruded.

(4) Prevention of further intrusion

The RCs at B, C, and D are activated to adjust the network. Based on security policy, Alice's accounts on database and machine C are closed. The administrator is informed to check the NFS on D.

5. Conclusions

In this paper we have presented an approach to detection, assessment, and prevention of further intrusions of distributed intrusions across a computer network. It has been shown that SDR is very useful for improving network security.

However, it is not easy to identify SDRs in a large network. In reality hackers may take advantage of any possible resources to attack a computer network, and these resources may be unstable and very difficult to categorize. These difficulties can reduce the effectiveness of our approach.

In addition, because our approach is profile-based, it has the disadvantage that it cannot detect intrusions which do not have identifiable anomalies. It is also difficult to find proper threshold values for intrusion detection and assessment.

References:

[1] B. Mukherjee and L. T. Heberlein, and K. N. Levitt, "Network Intrusion Detection", *IEEE Network*, pp. 26-41, May/June 1994.

[2] P. A. Porras and P. G. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances", *Proc. 1997 National Information Systems Security Conf., MD, October 1997*, available at http://www.sdl.sri.com/emerald/downloads-intmain.html

[3] G. Vigna and R. A. Kemmerer, "NetSTAT: A Network-based Intrusion Detection Approach", *Proc. 14th Annual Computer Security Applications Conf.*, December 1998, available at http://www.computer.org/conferen/proceed/acsac/8789/8789toc.htm

[4] P. Schnaidt, *Enterprise-wide networking*, Carmel, Ind., 1992.

[5] S. S. Yau and J. Zhu, "Intrusion Ripple Analysis in Distributed Information Systems", *Proc. 6th IEEE Workshop on Future Trends of Distributed Computing Systems*, October 1997, pp. 28-34.

[6] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, etc., "An Architecture for Intrusion Detection Using Autonomous Agents", *Proc. 14th Annual Computer Security Applications Conf.*, December, 1998, available at http://www.computer.org/conferen/proceed/acsac/8789/8789toc.htm

[7] G. B. White, E. A. Fisch, and U. W. Pooch, *Computer System and Network Security*, CRC Press, Inc. 1996.

[8] S. S. Yau and B. Xia, "An Approach to Distributed Component-Based Real-Time Application Software Development", *Proc.1st IEEE Int'l Symp. on Object-Oriented Real-Time Distributed Computing*, April 1998, pp. 275-283.

[9] H. S. Javitz and A. Valdes, "The SRI IDES Statistical Anomaly Detector", *Proc. IEEE Symp. on Security and Privacy*, May 1991, available at http://www2.csl.sri.com/intrusion/index.html

[10] D. Anderson, T. F. Lunt, etc., *Detecting Unusual Program Behavior Using the Statistical Components of NIDES*, SRI-CSL-95-06, May 1995, available at http://www.sdl.sri.com/nides/index5.html