Computer Science Masters Project

DIG

Detecting Intrusions Graphically

By

Christopher Cramer

Approved by:

Project Advisor : _____ Date: _____

Sponsoring Faculty : _____ Date: _____

Intrusion Detection

Network security has become a serious topic for research. With companies becoming increasingly dependent on the use of computers for managing resources the need for security remains critical to these industries. There are currently many systems available for network administrators to use to monitor their systems, including the DOORS system developed at RPI (1,2,3). Many of these systems use different tactics to protect systems. The novel idea that we are currently proposing involves using the characteristics of a user to determine when someone is attempting to gain illegal or unusual access to files. A good example of related work is the product NIDES from SRI international (4), or network (5) and host systems (6) developed at RPI. SRI actually has a history of research in the Intrusion Detection field.

There are many types of attacks a system might face, ranging from simple denial of service attacks (DOS) to complex attacks from multiple machines. Many of the systems that are available today are perhaps good at detecting one type of attack while failing at others. To date there has not been the production of a sure fired way to detect

intrusion. There are many reasons for this since there are so many possible methods to attack or damage a system. We feel that through the combination of many techniques it may be possible to improve the overall security of the system. Rather than attempt to build a mystical system that can detect and defend every attack, which would be unrealistic, we have decided to look into a subsection.

The attack that this system is primarily being designed for is the classic 'user to root'. This is where a user tries to get additional privileges such as access to log files or the ability to masquerade as another user. We would like to know if the user is trying to misuse the system. Once a user has gained 'root' access they generally have the ability to go in and change log files in order to 'cover their tracks'. Therefore the need to determine if a user is attempting to gain root access needs to be closely monitored.

A method of tracking users and how they use the system that they are on is by inspecting audit files. Auditing is the process by which records are maintained of a users activity on a system, and typically cannot be readily accessed for editing by anyone but the super-user or chief security officer. Through reconstructing these audit logs it becomes possible to trace what the user did. One of the benefits of an auditing program is that a system can be designed and created which can run real-time to detect intruders. This becomes very significant for a researcher or system administrator who wants to know as soon as misuse of their system occurs. The development of real-time intrusion detection is the purpose of this program.

To understand the information gathered from the auditing data, we must first understand what the audit data can or will give us. The system we are designing does not depend entirely on the format of the auditing data, but rather the information that can be

recovered. This will allow the system to be applied to a variety of operating systems. The first model auditing system that was used was data from the Basic Security Module gathered by DARPA. (7)

One of the best ways for humans to understand data is through visualizing it.(8) The first phase of this project is in creating a visualization tool for user data. The data to be represented is in a format, which may differ depending on the operating system. The program is written in JAVA so that it may be ported and initiated on many operating systems.

There are two complementary approaches to detecting intrusions, knowledge-based (AKA misuse detection) approaches and behavior-based approaches. Almost all ID systems today are knowledge-based. The system that we are designing is the rarer behavior based where we want to see when a user is acting differently from there 'typical' behavior.

Behavior based approaches to intrusion are limited by the assumption that a deviation from that users 'typical' behavior is a sign of an intrusion. The determination of what a user 'typical' behavior is may vary from individual to individual. For example, a typical user would not have the same patterns of using the file system as the super-user may have. The system is basically paranoid about anything that it has not seen before.

The strength of a system designed like this is that unknown and new attacks may be detected. The major drawback to this type of system is that many false alarms may be

triggered during the learning phase. There also remains the issue of updating the system if the behaviors change through time. (9)

Knowledge-based intrusion detection is based upon known attacks and weaknesses of systems. Any activity that is not previously known as threatening or malicious is deemed acceptable. Therefore, effectiveness depends on updated knowledge of possible attacks. One advantage of these systems is the low false alarm rate, but that is offset by the fact that they are defenseless against new or novel attacks. (10)

The Human Factor

It is a well-known fact that humans are adept at analyzing graphical data and determining patterns (11). Many tools that are common in many of the systems available on the market today include the notification of the super-user once someone has already broken in. Would it not be nice for an administrator to look at a session of a user and predict that they were going to commit a misuse of the system? The production of a graphical tool for the administrator to analyze user activity would be helpful. In addition the development of rules for a system may also arise from the simple inspection of known attack signatures.

The Intrusion Detection system that is being proposed could maintain use of such a GUI for monitoring. Many other items are typically being monitored such as bandwidth and disk space so why not create a program, which could watch what the user was doing. The problem concerning Privacy issues can be overcome by displaying not the actual commands the user was typing or the files they were accessing but by

displaying the classes they belong to. For example, the system administrator might see that there repeated attempts to access a file, which was denied, or repeatedly attempt connections to services that failed. The system administrator could then approach the individual or suspend the account until the matter was settled.

The Goals

The main goal of this project is to produce a tool that can be used in intrusion detection. The primary focus will be on the development of a graphical interface that can be used by a system administrator to monitor users behavior. That data that will be graphed comes from audit logs generated by the Basic Security Module a.k.a. BSM running on a Solaris machine. These logs can easily be generated on any Solaris system that has the audit daemon. Typically, the super-user can only run auditing, but this system uses the files, which are generated, so that any user who has permissions to read these files gains access to using the system. The system that is being developed was initially designed for Solaris but can be used on any system that uses auditing. Creating a system that can be used on different operating systems is a more useful than a proprietary one. Allowing flexibility will give other researchers the ability to use system in the future. Also the system was designed and implemented to act as a generic token filter, meaning that it could just as easily look at IP packets in a network.

The Files

The files that are included in this projected include the following:

- Filter_Data.java

- graph_app.java
- graph_data.java
- Id_Data.java
- Draw_Graph.java
- MyJButton.java
- Alphabetize_Data.java
- My_FileFilter.java
- Make_Independent_Frames_Graph.java
- Extract_Time.java
- Show_Graph_From_File.java

How To

This section describes how to set up your system to use the Intrusion Detection software that has been written. Typically audit files need to be collected from the system you are going to monitor. To collect these files on your system you will probably need to have root access to start the audit deamon process. The files that we will be using as examples were generated by the DARPA intrusion detection project. If needed a person who wishes to use this software can simply gather that audit files from an outside source or from the system administer, providing that they are willing to give access to you.

The program itself is written in JAVA so that it may be ported to numerous systems. A user manual accompanies the actual class files that are needed to run the application. The software that was designed for this system can be broke into two functionally different sections. The first section involves the processing of the raw data from the audit deamon. The second section deals with the graphical representation that is generated.

The program begins by prompting the user to choose which character is going to be the delimiter for the files they want to parse. Typical choices include a comma, space, or tab character. These are the most common types of 'delimited' files, however any delimiter may be choosen with the exception of the newline (\n).

The next step in the program is to the allow the user to choose a list of tokens to look for. The user does not have to choose a list of predefined tokens to search for but it will greatly increase effectiveness if they save a list of common tokens they typically are interested in. For example, in the BSM audit data it is very common to look for 'chdir' commands, rather than have to add these tokens in each time the user can save them in a file for later use.

Once the user has chosen which tokens they are interested in looking at then the main controlling frame of the application becomes present. This frame has buttons which control the other aspects of the program. From here the user can

1) add items to the list of tokens
2) open a previously filtered file
3) filter a new file
4) save the list of tokens

Adding a token item

To add an item to the list of tokens they simply type the text of the token they wish to search for, along with what value to assign to this token. The third text field that the must enter is a token location for searching at a specific token in the line. After they have entered the info a window will appear which will allow them to choose a color to be used for graphing the item.

Opening a filtered file

Once a file has been filtered the user has an option for saving this filter information in a file. This can be useful because of the great speed increase that is evident. Using this method for viewing a file means that the file has already been filtered and will reduce the processing time.

Filter a new file

This is the item that will be chosen the most often by the user. Here they pick a file that they would like to filter. The filtering begins with whatever tokens are in the current list. Since this is where the filtering occurs there will be a slight time delay depending on the size of the file and the number of tokens it is looking for. The graphical

representation of the data will then be displayed. The user then has the option of saving this file for viewing at a later time.

Save the token list

This will save whatever tokens are listed into a file which can be later read when the program starts.

Related Work

The GrIDS (Graph based Intrusion Detecion System)(12) is a good example of some of the software that is already available. This system creates graphs based on interaction between computer networks. Designed for detecting widespread attacks, like 'worms', like the infamous Internet Worm of 1988 (13).

The other great example of similar work is being done by SRI International, known as NIDES (4). This involves creating profiles for users to determine when their behavior differs from their typical behavior. Each user has behavior unique to their habits, by measuring multiple facets of this behavior it is possible to detect intrusion. Some of the measures the NIDES uses include:

* Ordinal measure - e.g. CPU utilization
* Categorical measure - frequency of a certain set of categories
* Binary Categorical measure - for tracking rarely used activities i.e. password changing
* Linear Categorical measure - count number of times for each category (how many 'ls' how many 'cd' commands)

This provides a great step toward gathering information regarding each user's typical activities. The major drawback to this is that the system must be modified over time as the users habits change. The intrusion detection goal, higher system security, may use this information to 'build a better mousetrap'. We plan to use a similar yet different strategy for our system, which will include the use of temporal information to detect intrusion.

Results:

The following results were obtained through the use of the DARPA data. Here a few of the graphs that were produced.

These results provide a nice graphic interface for an Intrusion Detection system, but merely comprise a small faction of such a system. It is the goal of this research to produce a tool, which may be used by others to further their research as well. The next logical step in a system of this nature is to design methods to determine intrusion through some type of data analysis rather than a human. Numerous options that are now being discussed by our group with regard to which direction to proceed in. One of the strongest considers time as the important factor in recognizing a person who is not behaving in their typical fashion.

Bibliography

(1) A. Bivens, L. Gao, M. F. Hulber and B.K. Szymanski, ``Agent-Based Network Monitoring," *Proc. Autonomous Agents99Conference,* Seattle, WA, May 1999, pp.

41-53.

(2) A. Bivens, P. Fry, L. Gao, M.F. Hulber, Q. Zhang and B.K. Szymanski, ``Distributed Object-Oriented Repositories for Network Management," *Proc. 13th Int. Conference on System Engineering*, pp. CS169-174, Las Vegas, NV, August, 1999.

(3) A. Bivens, R. Gupta, I. McLead, B. Szymanski and J. White, ``Scalability and Performance of an Agent-based Network Management Middleware," *International Journal of Network Management*, submitted, 2002.

(4) SRI International, "Next-generation intrusion detection expert system (NIDES): a summary," *Technical Report* SRI-CSL-95-07, Computer Science Laboratory, SRI International, Menlo Park, CA, May 1995.

(5) A. Bivens, M. Embrechts, C. Palagiri, R. Smith, and B.K. Szymanski, ``Network-based Intrusion Detection using Neural Networks," *Intelligent Engineering Systems through Artificial Neural Networks*, Vol. 12, Proc. ANNIE 2002 Conference, November 10-13, 2002, St. Louis, MI, ASME Press, New York, NY, 2002, pp. 579-584.

(6) J. Branch, A. Bivens, C-Y. Chan, T-K. Lee and B.K. Szymanski, 2002, ``Denial of Service Intrusion Detection Using Time Dependent Deterministic Finite Automata," *Proc. Graduate Research Conference*, Troy, NY, pp. 45-51.

(7) Wenke Lee Dong Xiang, "Information-Theoretic Measures for Anomaly Detection," 1999.

(8) Steven Hofmyer, Stephanie Forrest, and Anil Somayaji , "Intrusion Detection using a Sequence of System calls" *Technical Report*, University of New Mexico, 1998.

(9) SunSoft, *SunSHIELD Basic Security Model Guide,* SunSoft, Mountain View, CA, 1995.

(10) Kenneth C. Cox, Stephen G. Eick, and Graham J. Wills, "Visual Data Mining: Recognizing Telephone Calling Fraud," 1997, *Technical Report,* Bell Labs / Lucent Technologies.

(11) S. Staniford-Chen, S Cheung et al., "GrIDS: Graph Based Intrusion Detection For Large Network Systems," 1999, *Proceedings of the 19th National Information Systems Security Conference.*

(12) M. Eichin and J. Rochis, "With microscope and tweezers: An analysis of the Internet Worm of November 1988", *IEEE Symposium on Research in Security and Privacy*,1989