A NETWORK SECURITY MONITOR

L. Todd Heberlein, Gihan V. Dias, Karl N. Levitt, Biswanath Mukherjee, Jeff Wood and *David Wolber*

Division of Computer Science
Department of Electrical Engineering and Computer Science
University of California, Davis
Davis, CA 95616

Abstract

The study of security in computer networks is a rapidly growing area of interest because of the proliferation of networks and the paucity of security measures in most current networks. Since most networks consist of a collection of inter-connected local area networks (LANs), this paper concentrates on the security-related issues in a single broadcast LAN such as Ethernet. We formalize various possible network attacks. Our basic strategy is to develop profiles of usage of network resources and then compare current usage patterns with the historical profile to determine possible security violations. Thus, our work is similar to the host-based intrusion-detection systems such as SRI's IDES [9]. Different from such systems, however, is our use of a hierarchical model to refine the focus of the intrusion-detection mechanism. We also report on the development of our experimental LAN monitor currently under implementation. Several network attacks have been simulated and results on how the monitor has been able to detect these attacks are also analyzed. Initial results demonstrate that many network attacks are detectable with our monitor, although it can surely be defeated. Current work is focusing on the integration of network monitoring with host-based techniques.

1 INTRODUCTION

The study of security in computer networks is a rapidly growing area of interest [6, 7, 21]. This activity has been fueled by several recent network attacks (or network intrusions). The task of providing and maintaining security in a network is a particularly challenging one because of the following facts. First, there is a proliferation of local area networks (LANs) in academic, business, and research institutions, and these LANs are in turn interconnected with the "outside world" via gateways and wide area networks (WANs). Second, these networks and their associated equipment (including LANs, WANs, and gateways), when they were developed, were done so with trusted users in mind; the issue was to solve the networking problem and very few, if any, security measures were instituted. Consequently, network attacks or intrusions such as eavesdropping on information meant for someone else, illegally accessing information remotely, breaking into computers remotely, inserting erroneous information into files and flooding the network thereby reducing its effective channel capacity are not uncommon (see, for example, [17]).

To overcome these problems, several proposals suggest the deployment of *new*, *secure*, and possibly *closed* systems by using methods that can prevent network attacks, e.g., by using encryption techniques [13, 14, 16, 18, 20]. But we recognize that these solutions will not work because of the tremendous investment already made in the existing infrastructure of *open* data networks, however insecure the latter might be. Furthermore, encryption techniques cannot protect against stolen keys or legitimate users misusing their privileges. Hence, we approach the problem from a different angle. Specifically, our goal is to develop monitoring techniques that will enable us to maintain information of *normal* network activity (including those of the network's individual nodes, their users, their offered services, etc.) The monitor will be capable of *observing* current network activity, which, when compared with historical behavior, will enable it to detect in real-time possible security violations on the network – regardless of the network type, organization, and topology. Since our goal is to detect network intrusions, note that we are borrowing some of the basic concepts that have been developed or proposed for non-networked, stand-alone, intrusion-detection systems, e.g., IDES [2, 9], MIDAS [22], and others [10]. See [10] for a survey of intrusion-detection development efforts.

The focus of our present activity is narrowed to the local environment. In particular, we are developing our concepts for an Ethernet – Carrier Sense Multiple Access with Collision Detection (CSMA/CD) [11] – LAN which, because of its broadcast property, enables us to design and test a single secure monitor that has access to all of the network traffic. (Distributed monitoring of wide area networks will be considerably more complex, and will be taken up after our LAN monitoring problems have been properly tackled.) A prototype LAN security monitor – hereafter referred to as our Network Security Monitor (NSM) – has been in operation for over a year, and it is continuously being upgraded as we incorporate into it newer concepts as they emerge. The NSM in its most elementary (lowest) level of operation can measure network utilization and host-to-host activity. But when it suspects a possible intrusion or under the control of a Security Officer, it can also refine its focus on an individual user, a group of users, individual or group(s) of services they are using, etc., in a hierarchical fashion. Probabilistic, rule-based, and mixed approaches are being employed by the monitor, and it raises alarms for the Security Officer upon detecting anomalous behavior. The Security Officer interfaces with the monitor via a user-friendly window system, using which he/she can manually alter (usually refine) the monitor's focus as well. At present, the monitor is being employed to study network behavior and possible intrusions, and we report on them later in the paper.

The system model is described in the next section. We model network attacks in Section 3. The conceptual view of the NSM is developed in Section 4, and its details are provided in Section 5. Results from simulated attacks are analyzed in Section 6. We conclude in Section 7 by summarizing the paper and discussing future work.

2 SYSTEM MODEL

The system's operating environment, viz. the setting in which the NSM is deployed, is outlined below. Also included is our view of a network attack or intrusion [3, 12].

The target system, which needs to be protected from attack, consists of a number of host computers (including devices such as file servers, name servers, printers, etc.) and a LAN through which the hosts are inter-connected. The LAN is assumed to employ a broadcast medium (e.g., Ethernet), and all packets transmitted over the LAN are potentially available to any device connected on the network. The LAN is also assumed to be physically secure, in the sense that an attacker (intruder) will not be able to directly access the network hardware such as the connecting medium (cable) and the network interface at each host. The LAN is connected to the outside world via one or more gateways.

The principal source of attacks is assumed to originate from the outside world and not from a source which already has legitimate access to a host or the LAN. However, an intruder's strategy could be to initially infiltrate a less secure host on the LAN and then utilize this trust as a platform for launching the attack on the ultimate (main) target.

Of course, the most effective way of preventing attacks is to isolate the system from the outside world. However, there are many environments, which, while requiring that the integrity of the system is protected, need to operate in an open environment, as outlined below. First, the system may need to communicate with systems not controlled by its owners, and such systems, and the communication paths to them, may not necessarily be trusted. This communication can consists of user data (e.g., mail) and system data (name and file service, authentication, etc.) Second, the system may need to be built using off-the-shelf hardware and software, which may have (known or unknown) security problems. Finally, the system must use existing communications protocols.

In summary, the operating environment is modeled by the following: hosts, LAN (the wire, bridges, routers, gateways), and the outside world (viz. connections via gateways).

3 ATTACKS ON NETWORKED COMPUTERS

The sources of network attacks could be hosts on the LAN, devices connected to the LAN (e.g., wiretaps), and devices outside the LAN connected via a gateway. If the system owners have taken sufficient precautions regarding physical access to the hosts and the LAN, and regarding screening of users authorized to use the system, the remaining point of weakness is from outside the LAN. The targets of attacks could be hosts, the LAN (including bridges and gateways), and resources outside the LAN used by the system or its users.

An attacker can have a wide range of possible objectives. An attacker could be malicious (i.e., eager to cause harm), or benign (i.e., causing harm to the computer system, its owners, users, or usees is not his intention). However, the attacker could harm the system inadvertently. The objectives of an attacker could include: access the system "for fun"; use computing resources (CPU, disk, I/O devices, etc.) for his own purposes; obtain information stored on the system; modify or destroy information on the system; prevent or impede normal operation of the system; or damage or destroy the system.

An attack could be considered to be comprised of three phases, viz. preparation, execution, and post-attack. In the preparation phase, the attacker gathers information needed to launch the attack. The actual attack occurs in the execution phase. In the post-attack phase, the desired effects (including side effects) of the attack are observable. The three phases are analyzed in further detail in the following subsections.

3.1 The Preparation Phase

The effectiveness of an attacker, both in term of how far he can penetrate the system and how well he can avoid detection, depends to a large extent on how well-informed he is. The corresponding information is of two types – generic information such as break-in methods, common passwords, and weaknesses in operating systems; and specific information about the system to be attacked such as the number, types and names of hosts, the network configuration, the software (both system and applications) being run, users, their work patterns, and personal information about them (useful for guessing passwords), and information about sensitive data on the system.

A competent attacker is expected to have the generic information. However, he also needs the system-specific information. While there are a number of ways of obtaining such information (phone books, drivers license information, inside contacts, etc.), the network itself is a fruitful source of such information. Some utilities which provide a wealth of information in the Internet environment are: The Domain Naming System, NICname/whois service, Finger, Ruptime/rwho, and Sendmail. Details of these services and how they can be detected are discussed in Section 4.

3.2 The Attack Phase

Assume an attacker A which may be a hostile program or a human sitting at a computer. A wishes to attack a target T. In order to do so, A must establish a channel of communication with T. This may be done by A and T communicating directly with each other (for purposes of this discussion, a network operating as intended is considered simply as a communications channel and not an intermediary) or via an intermediary I, where A communicates with I and I communicates with T. An example of using an intermediary would be to remotely log in to a machine and then access another machine from it. For example, if a network component such as a gateway were subverted and made to perform differently than intended, then it would be considered an intermediary. In general, there could be n intermediaries, where n>=0.

Consider such a chain A - I(1) - I(2) - ... - I(n) - T. This implies that the attacker has obtained some measure of control over A and the I's and is using them to launch an attack on T. However, A must have launched an attack on I(n) from A and I(1), I(2), ... , I(n-1). Therefore, we see that an attack using a chain of intermediaries can be decomposed into a series of attacks, each of which adds to the set of entities under control of the attacker. For simplicity, we consider A and all of the I's together and refer to the composite group as A. Then, the attack simplifies to an attack from A to T, where A is a set of entities rather than a single entity.

For A and T to communicate, T must either *offer a service* which can be exploited by A, or T must seek to *use a service* offered by A. A may get T to use a service controlled by it by either obtaining control over a legitimate service provider or by impersonating one.

3.2.1 Services offered by hosts.

The lowest level of service provided over the network by hosts is the receiving and sending of packets. At the Ethernet and IP levels, hosts may accept, reject, or forward packets based on their source and destination addresses, protocol types, and other characteristics such as security options. Examples of higher-level services are *remote login, finger*, and *network file systems.* Securitywise, services can be ranked on two criteria, viz. the *degree of control* over the system given by the service, and the *strength of the authentication* performed. Ideally, as the degree of control increases, so should the strength of the authentication.

3.2.2 Services offered by network.

The primary service offered by a network (including gateways, etc.) is the transmission of packets. Other services offered are the routing of packets and response to network management commands. These services too can be ranked on the degree of control provided and on the authentication required.

3.2.3 Services used by hosts.

Hosts use the services provided by the network to send and receive packets and the services provided by other hosts such as resource location, network file systems, etc. In this case, a host is vulnerable to incorrect information being provided by the service. For example, a resource locator may return the identity of a resource controlled by the attacker. The purpose of authentication in this case is to ensure the legitimacy of the information being provided.

3.2.4 How attackers may exploit services.

An attacker may utilize a service in two ways. First, the service, as documented and intended to operate, may contain security holes and weaknesses. These may be compounded by poor operating practices of users and system administrators, e.g., poor choice of passwords. Second, due to bugs and trapdoors, the implementation of the service may allow attackers to use the service in ways not intended by the designers. (Note that there is sometimes a fine line between bugs and features!) For example, in some operating systems, hitting an interrupt character before the login authentication is completed will allow a login without a password, and some operating systems will crash the host when certain types of Ethernet packets are received. See [5] for examples of some services offered and used by BSD Unix together with what they allow a user to do, and the type of authentication performed.

3.3 The Post-Attack Phase

A system may continue to exhibit changes even after the activity of the attack is over. This may consist of the effects desired by the attacker and possible side effects. From the point of view of the system owner, effects of an attack could include the following.

- Dissemination of data stored on the system.
- Loss or reduction of system services, possibly due to the attacker's use of services or by the attacker causing damage to the system.
- Loss of system integrity and confidence in the system. Once a system has been penetrated, there is always a possibility that the attacker may do so again, possibly via trapdoors left open the first time.

In [5], we survey several methods for detecting intrusions that employ services such as *Whois / Finger, Mail / SMTP, Remote Login, Network File Systems*, and *Domain Name Service (DNS)*, or perform *misrouting of network traffic* and *overloading the system.*

4 CONCEPT OF THE N.S.M.

This section presents the conceptual view of the NSM. Currently, the NSM uses a four dimensional matrix of which the axes are: Source (a host which generates traffic), Destination (a host to which traffic is destined), Service (mail, login, etc.), and Connection ID (a unique identifier for a specific connection). Each cell in the matrix represents a unique connection on the network from a source host to a destination host by a specific service. This matrix is similar in concept to the well-known access matrix, the basis for protection in many systems. Each cell holds two values: the number of packets passed on the connection for a certain time interval, and the sum of the data carried by those packets. An analyzer must examine the data patterns in the matrix representing the current traffic to determine if an attack is occuring on the system.

One method to examine the traffic matrix is to compare it against a matrix holding a certain pattern. For example, a comparison may be made against a matrix holding the representation of a specific attack. To compare the two matrices, the pattern being checked can be treated as a mask through which the current traffic will be passed. A mask is a representation of the values for traffic measurements which are observed for some known traffic pattern. When the current traffic measurements are presented to the mask, only the measurements which have values matching the mask will pass through. A high percentage of the measurements passing through the mask would indicate that the traffic pattern for the mask is occurring.

Many problems which can occur in a networked environment, from a simple node going down to a network worm, can be used to generate a mask. However, acquiring these masks can be difficult since there are simply not enough examples to generate many of the desired masks. Simulations could also possibly be used to generate attack patterns. However, we would only be generating masks for predetermined problems or attacks. An original problem or attack could go unnoticed.

Therefore, for the present time, we have followed a similar path as that by IDES [9] and Wisdom and Sense [19], viz. we generate a mask of the normal traffic and detect anything outside this pattern. This is based on the Denning model [2] which assumes that an attack would generate anomalous patterns. For this approach, we must treat our mask in just the opposite way as before. Our mask now only allows measurements which do not match this "normal" mask to pass through. (See Fig. 1.)

The actual representation of the mask can be performed by several methods. We currently generate a probabilistic distribution of values for each measurement which is seen frequently. The value for a current measurement is compared to the distribution, and if the probability of that value occurring is too low, it is passed through the mask. Although we have had excellent results with this method, it is very expensive in terms of memory requirement, and it may not be appropriate in all environments. We are currently looking at generating masks using the techniques used by IDES, Wisdom and Sense, and other intrusion detection systems, so that we can make an experimental analysis of these different methods.

The NSM is designed to operate in an open environment in real time. To this end, any analysis to be done must match the capabilities of the machine as well as the amount of traffic which is occurring. The matrices generated by the NSM, even a sparse one, can contain a very large number of measurements to be examined. A trade-off must often be made as to how much analysis is performed on measurements and how frequently this analysis is to be performed. The NSM, therefore, groups cells in a logical and hierarchical fashion. The groups are then presented to a mask, which in turn has been grouped. If a group passes through the mask, this group can be presented to the security officer; furthermore, the NSM can break the group into the smaller constituents to perform a more detailed analysis. (See Figs. 2 and 3.)

Our groupings are based on the axes of the matrix. Each level of grouping effectively reduces the dimension of our matrix by one. All the connections of a specific service between two hosts are grouped into a "Source-Destination-Service" group representing all the traffic flowing from the Source to the Destination by that Service. Each of the service groups for a pair of hosts are then grouped into a "Source-Destination" group representing all the traffic flowing from the Source to the Destination. All the "Source-Destination" groups for a specific source host are grouped into a "Source" group representing all the traffic generated by that Source. The result is a hierarchical structuring of groups from the Source group to the individual cell.

This hierarchical structuring allows for a monte carlo divide-and-conquer search of the entire network traffic. We are able to perform fast analysis required for real time analysis, but we can only be sure with a high probability of detecting an attack. If processing power is available, greater analysis may be conducted on groups which do not show abnormality, in order to reduce the chances that the probabilistic search presented an incorrect answer.

Other structured groupings may also be desired. Examples include grouping services which use a particular implementation, grouping services by the level of authentication they require, grouping hosts by the operating systems they use, and grouping hosts by their physical location.

The second method to examine the current traffic matrix is to apply a set of rules against the matrix. This method is particularly important if profile masks have yet to be generated. Since the rules look for specific traffic patterns, they can be transformed into matrix masks too; therefore, only the single analysis tool, passing current traffic through masks, needs to be used. Unfortunately, after examining a number of potential rules, we have found not all rules apply well at all grouping levels, so a mask may only be applicable at a single level. For example, a rule looking for a login connection which only exchanges a few packets and terminates (thus indicating a possible failed login) does not map well to the Source-Destination group level. Conversely, a rule looking for a host communicating with a large number of other hosts works well at the Source-Destination level, but it does not work well at the connection level.

5 DETAILS OF THE N.S.M.

This section examines the details of the NSM prototype. The NSM was built on a Sun-3/50 workstation and it consists of five separate components: a packet catcher, a parser, a matrix generator, a matrix analyzer, and a matrix archiver. A description of these basic components and of the overall system is given, followed by a more in-depth examination of the matrix analyzer component. A description of an interface to the system, which is under construction, is presented at the end of this section.

5.1 Overall Structure

The NSM prototype consists of the five main components linked in a pipeline fashion. The components are modular so they may be modified separately, as long as their interfaces remain unchanged. They may also be used as parts of other programs – the first three components are used by another of our projects, the Eavesdropper, and components one, two, three, and five were used to generate the data to build the profile.

The *packet catcher* captures the traffic off the network, collects the individual bits into separate Ethernet packets, and passes each packet to the parser. Of the five components of the NSM, this is the only one that is platform-dependent. It must be able to put the Ethernet hardware into promiscuous mode, so that all traffic, not just the traffic destined for the host on which the monitor is running, is captured.

The *parser* takes the packet from the packet catcher, parses the layers of protocol, extracts pertinent information from each layer, and passes the information to the matrix generator. The parser needs to have detailed knowledge of the protocols it is required to parse. The pertinent information consists of the packet's source, the packet's destination, the service, which host initiated the connection, and a unique thread ID. Although we are currently only parsing IP and TCP protocols, this pertinent information should be available in most other protocols as well.

The *matrix generator* takes the information passed down from the parser, finds a cell in the Access Control Matrix, or current traffic matrix, to which the packet belongs, and increments a counter in that cell. Each cell in the matrix represents a single connection across the network. The counter in the cell indicates how many packets have been generated by this single connection. A counter also resides in the cell to indicate how many bytes of data have been generated by the connection (a packet may contain a variable amount of data), but is not used in the prototype. This matrix location is based on the 4-tuple <source, destination, service, connection ID>. A static matrix to hold every possible 4-tuple is prohibitively large (the source and destination fields are 32 bits long), so the sparse matrix is implemented with linked lists. This sparse matrix is shared with the matrix analyzer. In addition to communicating with the matrix analyzer by updating counters in the matrix cells, every time a new node has to be generated, a message is sent to the matrix analyzer to indicate that a new communication has begun.

The linked-list matrix format consists of a list of nodes containing the addresses of hosts which have placed a packet on the network. Each of these "source" nodes has a list of nodes holding the addresses of hosts to which it, the source node, has sent a packet. Each of these "destination" nodes has a list of nodes holding information about each service used between the source and destination hosts. Each "service" node has a list of nodes holding information about each connection using the service between the source and destination hosts. Finally, each "connection" node contains the number of packets used by the connection and which host, the source or destination, initiated the connection.

The source, destination, and service nodes also contain current information about the nodes below them. This corresponds to the grouping of cells mentioned previously. The service node contains the sum of all the packets using the service between the source and destination nodes, and the service node knows how many connection nodes are below it. The destination node contains the sum of all the packets which have passed between the source and the destination, and it knows how many service nodes are below it. Finally, the source node contains the sum of all the packets which it has generated, and it knows how many destination nodes are below it. Since the placement of each packet must go through each node along its path to the proper "connection" node, the nodes along the path to insertion simply increment a counter every time a search passes through it. Thus, no extra work is required to keep the aggregate totals.

The *matrix analyzer* examines the matrix representing the current traffic. The analysis is done by two different methods: examining the current network traffic against "normal" network traffic, the masking technique, and by applying rules to the current network traffic to look for specific patterns. The matrix analyzer is triggered by two different events: first, when a new node is generated by the matrix generator, a quick analysis is made of the new connection, and second, an alarm sounds at prescribed intervals to start a thorough analysis. The current monitor checks every five minutes. Theoretically, the matrix analyzer should do a thorough analysis continually. In practice, however, enough computing power may not be available, so a compromise must be made – we simply chose five minutes intervals. Furthermore, if a connection passes the initial quick analysis, a thorough analysis would not detect anything until enough packets have been generated to indicate something abnormal. After every third check, a message is sent to the matrix archiver to store the current matrix.

The matrix analyzer also handles the reporting of problems to a security officer. Eventually another component, the NSM user interface, will be added to generate a powerful but easy to use interface for the security officer. The matrix analyzer will then pass its results to the interface module which will determine how to present the results to the officer.

Finally, the *matrix archiver* writes the matrix representing the current traffic out to disk. Currently, a signal to save the matrix is sent to the archiver by the matrix analyzer every fifteen minutes. The size of our archive files is approximately two and a half kilobytes when compressed. Thus, approximately one megabyte of storage is used every four days. The archived files can be used to build or update a network profile. Also, if a previously unsuspicious host is marked as suspicious, its previous network activity can be tracked.

5.2 Analysis Phase

As indicated previously, the matrix analyzer examines the matrix representing the current traffic. Specifically, it looks for unusual traffic patterns and particular traffic patterns. Searching the traffic for unusual traffic requires knowledge of normal network activity which initially may not be available. Therefore, the specific traffic pattern detection scheme is essential.

To detect specific patterns in the network traffic, a series of rules is applied to the current matrix. These rules look for traffic patterns the author, the writer of the rules, imagines an attack will generate. The prototype is currently looking for very simple patterns: a single host communicating with more than fifteen other hosts, logins (or attempted logins) from one host to fifteen or more other hosts, and an any attempt to communicate with a non-existent host. These rules scan for unimaginative and systematic attempts to break into a local computer system. More elaborate rules may be easily (and are being) added.

Detecting unusual patterns by a probabilistic analysis of the traffic requires knowledge of what the normal traffic flow is. The current traffic matrix is then compared to the normal/abnormal traffic mask to determine if something unusual is happening. Analysis of the network traffic of our department Ethernet shows that most hosts communicate almost exclusively with a very small subset of other hosts (roughly three to five other hosts), and when these hosts do communicate, the same services are almost always used. Thus, even though there is a very large number of possible communication paths, only a very small subset is used. Any attack, even by local machines, would need to have intimate knowledge of these communication paths to go undetected.

The prototype examines the current network traffic when a new node is added, and at five minute intervals. When a new node is added to the network, the probabilistic analysis examines the cell against the normal/abnormal mask. At five minute intervals, the entire traffic matrix is compared to the normal/abnormal mask and the rules. Examining the probability that each path will exist and the probability that the amount of traffic generated on each path is normal can be expensive, so the hierarchical search pattern is used to limit the depth of the search. The search examines the summary information at each index node, the grouping information mentioned previously, to determine whether to perform an analysis deeper into the matrix. For example, if two nodes are communicating within normal boundaries, further examination of the individual services and connections may not be conducted.

Finally, the NSM's normal profile does not consist simply of a mean and variance; it consists of a range of values and the probability of observing a value at each range. Careful examination of network traffic showed that data amounts were not always Gaussian distributed; therefore, the mean and variance could not capture the true shape of the data.

5.3 NSM User Interface

The user interface for the NSM is under development. Its purpose is to provide a user (e.g., the security officer) information about the Access Control Matrix (ACM) in pictorial form that can be used to alert the security officer to attacks that change the tactics of the NSM.

In implementing the NSM interface, several goals had to be accomplished, and these goals are outlined below.

- **Real-Time Display.** This is accomplished through the use of a set of tools that gradually set constant elements of the ACM vector: (to-host,from-host,service). This process minimizes specific data requests to the ACM.
- **Ease of the Interface.** By working with the X-Window Interface environment coupled with the Athena Widget Toolkit, all tools are controlled through positioning of a mouse control with the keyboard regulated to customization preferences handling.
- **Readability of Displayed Data.** The three different tools graphically illustrate data such as 'blacked' hosts with no connections to other machines in the Variable Display; relative value boxes in the Grid Display for various measures; and actual information flowing between hosts in the Connection Display.
- **Portability of System.** The X-Window Interface is felt to maximize the possibility that the system will be portable due to its wide-spread influence in the computing industry.
- **Non-Competition with the Actual NSM.** Much of the work for display of the data is done through the use of the Toolkit and the Window servers, thereby freeing up the NSM to concentrate on its detection routine(s).

Future planned implementation of tools include various 'dials' and 'gauges' as extensions of the Grid Display tool, as well as a tool for interaction between the NSM and the security officer. Also needed and planned are the implementation of 'groups' to assign a common name to a group of hosts as well as increasing the vector to handle a user and time variable.

This interface shares many qualities with the IDES system interface, in that its purpose is to show the current state of traffic in a machine. The NSM, however, works on a larger scale than the IDES system, which is intended normally for single-host security analysis. The majority of this difference consists of the different priorities of what each interface reports. However, the IDES system currently has advantages that the NSM lacks such as aliases. Future NSM enhancements will remove this deficiency.

6 PERFORMANCE OF THE N.S.M.

We had several goals to accomplish with the early prototype of the NSM. Among these were the measurement of the actual data paths used for network traffic, an examination of the distribution of the values of the data for the different measurements, a determination of the processing power actually required by the NSM, and a determination of the types and numbers of problems reported. These information will be used for further research into the network-based intrusion detection method.

A data path is defined to be a means by which two hosts can communicate. This is generally provided by network services on the hosts. (Communication via removable media such as disks or tapes is not considered.) Thus, the total number of data paths between two hosts is defined to be the total number of network services by which the two hosts may communicate. Then, the total number of possible data paths is the number of possible host pairs multiplied by the number of services each pair can use.

We define a data path to be used if at least one connection was established on that path during a two-week observation period. Even with this very conservative definition, we found that only 0.6 percent of all of the possible data pathswere used. Therefore, our sparse matrix representing the normal traffic pattern needs to be only 0.6 percent of the possible matrix size. Furthermore, a random attack on this network would only have a 0.6 percent chance of using a data path which is normally used.

The distribution of the data was found to be generally multimodal (and not Gaussian). However, with many of the services, it is possible to mask out traffic generated automatically by the service itself or by the TCP protocol in order to arrive at a data distribution that is closer to Gaussian. This is important since many statistical techniques assume the data to be Gaussian distributed.

The prototype has been run on a Sun-3/50 workstation with no major drop in performance for other users on the machine. Only very simple analysis was performed, however (see Section 5.2). Reducing the threshold at which further analysis is performed in the matrix hierarchy can dramatically increase the amount of analysis required. Some training will be needed to determine the appropriate thresholds. Performing more complex analysis of anomalies (see below) can also dramatically increase CPU usage.

Many of the problems we detected on the network were simply abuse of network privileges. Problems such as frequent full back-ups of files over the network using FTP were common. Programs which continually executed finger programs were also discovered.

Some problems proved difficult to isolate with our simple analysis. For example, once we had a host which was a file server for several other hosts go down. Suddenly, we had many reports of problems from many different machines, and our first thought was that we had another worm. Had we known the relationship between these hosts, we could have come to a correct conclusion immediately. A network bridge going down also generated a large number of errors. Knowledge of the relationship between the different nodes on the network would be very useful in correlating problems on the network.

Knowledge of particular services can also be very important. For example, receiving mail from an unknown host should not cause as much concern as receiving a login from an unknown host. Similarly, many warnings were posted when a computer game called "Empire" was initiated on a local host, and the game and the host's address were announced on the local usenet network.

On a few occasions, a number of potential break-ins were detected. Several times, from one of our dial-up ports, large numbers of consecutive failed logins were detected. Also, periodically, some of our alumni would try to log in to their old accounts. Although their accounts were often still active, a login from an unknown host set off the monitor, requiring us to track down the problem.

The biggest concern was the detection of unusual activity which was not obviously an attack. Often we did not have someone to monitor the actual connection, and we often did not have any supporting evidence to prove or disprove that an attack had occurred. One possible solution would be to save the actual data crossing the connection, so that an exact recording of what had happened would exist. A second solution would be to examine audit trails generated by one of the hosts concerned. Both approaches are currently being examined.

In general, we were pleased with the performance of our preliminary NSM prototype. The matrix proved sparse, the hierarchical model did reduce the computational requirements to a point where real-time analysis could be performed, and we were able to detect several abuses and intrusions on our own departmental Ethernet network.

7 CONCLUSION AND FUTURE WORK

We have discussed an approach to obtaining network security based on capturing and analyzing network activity. The need for a security monitor is clear: most networks are intrinsically insecure as are the hosts that are attached to the network, and the network must be protected against users (insiders and outsiders) misusing privileges.

The paper establishes an implemented framework (called the NSM) for coping with network attacks. The NSM, working on an Ethernet, although most of the system is independent of the network type, captures and analyzes every packet in real time. An use of the network is considered suspicious if it is very dissimilar from previous uses (aka profiles) or if it is inconsistent with one or more policies. Similar methods for flagging attacks are the basis for host-based security monitors.

The network model offers the opportunity for a hierarchical analysis of activity. At the lowest level, host-to-host activity is analyzed; at the next level, it is services; and at the next level, it is connections. The lowest level is the first line of defense, passing suspicious behavior to the higher levels. This is the manner in which the NSM works autonomously. Under the security officer's control, the requests for data start at the top level and proceed downwards. Work is in progress on a more detailed analysis of network activity involving users and applications.

The paper also presents a model of network-based attacks, the model reflecting the phases of an attack, the services used, and the purpose of the attack. The attacks have a commonality, in that a user gains access to the network and then attempts to determine what the hosts can offer him or attempts to damage the network.

Many attacks will take this form, and will be detectable by the NSM in real-time. More subtle attacks will not leave so obvious a trail in network behavior. For example, an attacker could guess a password for a host, and use the *rcp* facility to copy the password file from another host for the ultimate purpose of cracking passwords. (Of course, the NSM could contain rules to be suspicious of the password file being transfered, but one could easily think of file names that would not be suspicious to the NSM.) Thus, a comprehensive monitor would also involve host-based monitors to watch over the activities of individual hosts. We are considering such hybrid systems.

Our initial results are promising and the overall framework for network monitoring allows integrating the NSM with the analysis software that is part of current host-based monitors. Clearly, however, it is essential to install the NSM (and other monitors) into real settings for extended times and determine their effectiveness in coping with real attacks.

Finally, we remark that our present network monitoring activities are confined to the local environment because the broadcast property of LANs enables us to design and test a single secure monitor that has access to all of the network traffic. Distributed monitoring of wide area networks will undoubtedly be more complex, and it will be taken up after our experience from LAN monitoring matures. In an irregular-structured, store-and-forward network, a single location of the monitor will no longer suffice since all network packets will not necessarily be routed through a particular node. Hence, the network monitoring functions have to be distributed among several nodes. These nodes will exchange information to reach a consensus on whether an attack is in progress. Noting that some of these nodes might have themselves been compromised, the distributed monitoring mechanism is expected to borrow some of the concepts from the Byzantine Generals Problem [8].

References

[1] S. M. Bellovin, "Security problems in the TCP/IP protocol suite," *Computer Communications Review*, vol. 19, April 1989.

[2] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engg.*, vol. SE-13, pp. 222-232, Feb. 1987.

[3] D. Estrin, "Controls for interoganization networks," *IEEE Transactions on Software Engineering*, vol. SE-13, Feb. 1987.

[4] F. T. Grampp and R. H. Morris, "Unix operating system security," *AT and T Bell Labs Technical Journal*, vol. 63, Oct. 1984.

[5] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, and J. Wood, "Network attacks and an Ethernet-based network security monitor," *Proc., 13th DOE Security Group Conf.*, Augusta, GA, May 1990, to appear.

[6] *IEEE Journal on Selected Areas in Communications*, Special issue on Secure Communications, vol. SAC-7, May 1989.

[7] *IEEE Network Magazine*, Special issue on Network Security, vol. 1, April 1987.

[8] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, July 1982.

[9] T. F. Lunt, et. al., "IDES: The enhanced prototype," Technical Report No. SRI-CSL-88-12, SRI International, Menlo Park, CA, Oct. 1988.

[10] T. F. Lunt, "Automated audit trail analysis and intrusion detection: A survey," *Proc., 11th National Computer Security Conf.*, Baltimore, MD, Oct. 1988.

[11] R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed packet switching for local computer networks," *Communications of the ACM*, July 1976.

[12] D. M. Nessett, "Factors affecting distributed system security," *IEEE Transactions on Software Engineering*, vol. SE-13, Feb. 1987.

[13] R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM*, vol. 21, Dec. 1978.

[14] D. B. Newman Jr., J. K. Omura, and R. L. Pickholtz, "Public key management for network security," *IEEE Network Magazine*, vol. 1, pp. 11-16, April 1987.

[15] D. B. Parker, *Fighting Computer Crime*, New York: Scribner's, 1983.

[16] L. Rutledge and L. Hoffman, "A survey of issues in computer network security," *Computers and Security*, vol. 5, pp. 296-308, 1986.

[17] C. Stoll, "Stalking the wily hacker," *Communications of the ACM*, vol. 31, pp. 484-497, May 1988.

[18] D. Tensa, "Typical weaknesses in operating systems software," *Information Age*, pp. 74-78, 1987.

[19] H. S. Vaccaro and G. E. Liepins, "Detection of anomalous computer session activity," *Proc., 1989 IEEE Symp. on Research in Security and Privacy*, Oakland, CA, pp. 280-289, May 1989.

[20] V. Voydock and S. Kent, "Security in high-level network protocols," *IEEE Communications Magazine*, pp. 12-24, July 1985.

[21] S. T. Walker, "Network security: The parts of the sum," *Proc., 1989 IEEE Symp. on Research in Security and Privacy*, Oakland, CA, pp. 2-8, May 1989.

[22] R. A. Whitehurst, "Expert systems in intrusion detection: A case study," Computer Science Lab., SRI International, Menlo Park, CA, Nov. 1987.

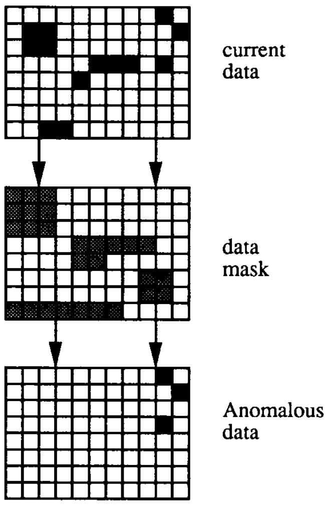

Fig. 1

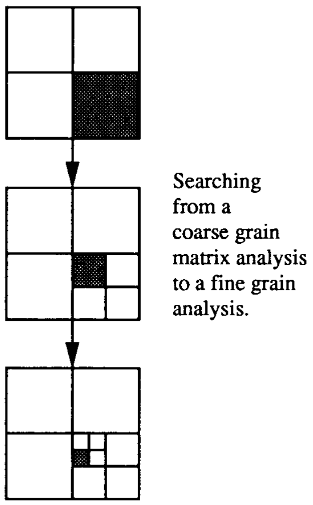

Fig. 3

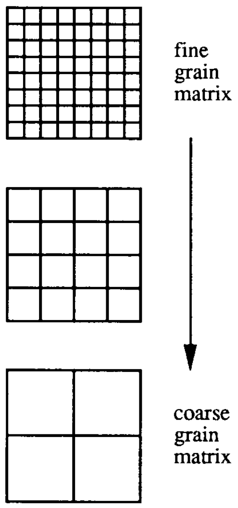

Fig. 2