UCRL CR 105095

A NETWORK SECURITY MONITOR

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

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

UCRL-CR--105095

DE91 007139

November 1989

PAN EX1016
Palo Alto Networks v. Centripetal Networks
IPR2022-00182
Page 00001

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

A NETWORK SECURITY MONITOR

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. Specifically, we formalize various possible network attacks and outline methods of detecting them. 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 [LUNT88a]. 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.

I. INTRODUCTION

The study of security in computer networks is a rapidly growing area of interest [NETW87, JSAC89, WALK89]. This activity has been fueled by several recent network attacks (or network intrusions). The task of providing and maintaining security in a network is 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, and flooding the network thereby reducing its effective channel capacity are not uncommon (see, for example, [STOL88]).

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 [NEWM87, NESH78, VOKE85, RUHO86, TENS87]. 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 violation 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 [DENN87, LUNT88a], MIDAS [WHIT87], and others [LUNT88b].

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) [MEBO76] — 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 approximately a year, and it is continuously been 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 *catch* several simulated network attacks, and we report on them later in the paper.

The system model is described in the next section. We model network attacks in Section III, and study their possible detection mechanisms in Section IV. The conceptual view of the NSM is developed in Section V, and its details are provided in Section VI. Results from simulated attacks are analyzed in Section VII. We conclude in Section VIII

by summarizing the paper and discussing future work.

II. SYSTEM MODEL

The system's operating environment, viz. the setting in which the NSM is deployed, is outlined below. Also included are pertinent definitions, e.g., our view of a network attack or intrusion [NESS87, ESTR87].

The target system, which needs to . rotected 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 attack 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 needs to communicate with systems not controlled by its owners, and such systems, and the communication paths to them, are not necessarily

trusted. This communication consists of user data (e.g., mail) and system data (name and file service, authentication, etc.) Second, the system needs 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, and bridges, routers, and gateways), and the outside world (viz. connections via gateways).

Finally, for the sake of this work, we adopt the following view of a network attack or intrusion. A network attack or intrusion is (1) an attempt (successful or unsuccessful) to use computer services (including but not limited to CPU, disk space, data, etc.) by an entity (person, organization, or process) which is not authorized to do so; or (2) use of computer services in such a manner so as to cause harm to the computer system, its owners, users, or usees.

III. 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 comprise of three phases, viz. preparation, execution, and post-attack (see Fig. 1). 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.

A. 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 IV.

B. 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 $n \geq 0$.

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.

(i) 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 given increases, so should the strength of the authentication.

(ii) 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.

(iii) 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.

(iv) 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.

(v) **Examples of services.** We give 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.

Service	Allows	Authentication
finger	information about users	none
mail	writing to mail file	essentially none
user FTP	read/write files	password
anonymous FTP	read/write restricted set of files	none
rlogin	log-in privileges	access control list / password
name service	name - address translation	host address
network file systems	read/write/execute files	host address

C. 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:

- 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.

IV. DETECTING AN ATTACK

The principal problem in detecting an attack is distinguishing it from normal system activity. Our approach is to rate activities on their likelihood of being an attack and concentrate on those deemed more likely to be an attack. The following criteria are used.

- Source of the message — Some sources, especially those outside the LAN or those with low intrinsic security (e.g., terminal servers, PC's, etc.) are more likely to launch an attack.
- Destination of the message — Both hosts which contain sensitive information (or are otherwise attractive to an attacker) and hosts with poor security (which can be used as intermediaries) are likely targets of an attack.
- Service used — Services with poor authentication, or services which yield more advantages to the attacker, are more susceptible.
- Contents — The contents of messages can be analyzed to determine their legitimacy. In general, this is hard, since contents of user messages tend to be unstructured and vary widely. Control messages used by various protocols (e.g., mail, the initialization part of rlogin) are well structured, and can be analyzed.

It could be expected that an attacker would attempt to make use of the network services described in Section III to obtain information to prepare for the attack. Therefore, we can detect such attempts by monitoring the network. Because there are many legitimate uses for such information, the majority of queries may not be indicative of an attack. However, excessive queries to such services or queries which appear to be gathering information and which would be useful to an attacker may be an indication that preparation for an attack is in progress.

Accessing system services from unusual locations, at unusual times, or with unusual patterns of activity may also be an indication of an attack in progress. In deciding

whether an observed activity is an attack, not only the documented features of the service but also possible bugs must be considered. In this respect, unusual or infrequently-used services may be regarded with more suspicion, since well-known services have been extensively analyzed and have stood the test of time so that most of their weaknesses can be expected to be known and possibly corrected as well.

Detecting that an attack has occurred by observing its effects can be done in two ways. The first is by analyzing system logs and audit trails for evidence of the attacker's actions, and the second is by observing changes to system behavior due to the attack. Examples of the latter are hosts crashing or not responding to network queries, or sending unusual types or numbers of messages. If sensitive data is tagged or can be otherwise identified by observing it when it is being sent across the network, it is possible to monitor network traffic for such data being read by an attacker.

In the following subsections, examples of some known methods of attack are analyzed [GRMO84, NEBE89, PARK83].

A. Eavesdropping

Eavesdropping on network traffic can be done from a device connected to the network medium. Eavesdropping on connections outside the organization is often easy because communication links are often not under their control. As well as obtaining sensitive data, eavesdropping can be used to obtain passwords which can later be used to log in to hosts.

Detection of passive eavesdropping is difficult, unless the physical equipment used for eavesdropping can be located. Physical security and encryption of data sent on insecure links can help prevent eavesdropping.

B. Whois / Finger

The *whois* and *finger* services provide information about users of the system. While the information provided is (or should be) non-sensitive, it could be used for compiling information about an organization (such as which department a person is in and the composition of project groups). It can also be used to gather information on account names and log-in patterns of users in preparation for an attack.

Detection of *finger* attacks is done by observing unusual patterns of activity. For instance, repeated fingering of the same host, or fingering of all the hosts on a network, may be considered suspicious.

C. Mail / SMTP

Mail is typically considered a write-only service. However, it is possible to perform an attack by sending a message which executes system commands when the recipient reads or executes it. Such a message is called a Trojan horse. In the Internet environment, mail currently does not allow authentication (although use of RFC 1113-1115 may change this) and it may be possible to get a mail recipient to take some action by sending forged mail.

The SMTP (Simple Mail Transfer Protocol) service performs other actions in addition to delivering mail. It can be used to ascertain the contents of mailing lists and verify user id's. Also, if its debug mode is enabled, it can be used to issue commands to the system.

Mail-based attacks can be detected by comparing the Internet source address with the source specified, and by monitoring the SMTP service to ensure that it is followed correctly. Also, mail connections can be monitored to ensure that only authorized gateway hosts send and receive mail from outside the organization.

D. Remote Login

Remote login, as provided by the BSD Unix *rlogin* service or similar services such as *rsh* or *telnet,* allows a user to give commands as if he were on a directly connected terminal, and it enables users to run arbitrary programs on a system from a remote location. Many systems allow system administrator (root) privileges by remote login. The *rlogin* command allows access to a system by giving a password or by being on an access control list (ACL) kept on the system. The ACL is a list of host and user names. In addition to logging in by ascertaining the password, an attacker could use the access control facility by

- Obtaining access to an account listed in the target's ACL.
- Obtaining access to the file the list is stored in (*.rhosts*) and modifying it.
- Subverting the name to address translation service to yield a false address for a host in the ACL.
- Subverting the network to make it appear the attacker is at a host address corresponding to a host named in the ACL.
- Obtaining system privileges on a host named in the ACL and masquerading as the user named in the ACL.

The above illustrates the fact that security is obtained from a chain of factors and the chain is only as strong as its weakest link. In normal use, remote logins tend to be from local hosts plus a limited number of remote hosts. Monitoring of the source address and the destination host and user is useful in detecting login attacks.

E. Network File Systems

Network file systems are services provided by hosts to other hosts on the network, and hosts are therefore susceptible to attack both as service providers and service users. In the first case, an attacker could access a network file service to read from and write to

files, and in the second, he could provide a file system containing bad data or programs to be used by the system under attack.

As in the case of login, detecting file system traffic to or from the outside could be a sign of an attack.

F. Misrouting Network Traffic

This is an example of an attack on a network and it can be carried out by sending fraudulent control messages to gateways, routers and hosts. Misrouting could be used to direct unauthorized traffic to an attacker's machine, where it could be examined and possibly modified, to masquerade as hosts, or to prevent hosts from communicating with each other.

Detection of attacks on network components is done by observing control packets used to modify routing tables, and by comparing routing tables in hosts and gateways with the expected values.

G. Overloading the System

A denial-of-service attack can be performed by overloading various parts of the system, such as hosts, the network, and gateways. One method of overloading a network is by simply sending a stream of packets which exceed the capacity of the network or a particular part of the network. Variations of this scheme, in which each packet sent causes a large number of packets to be generated at the target network, can be used to disrupt a network without overloading the sender. Another strategy could be to open a large number of connections to a target host or network. Since host and gateway routing tables have limited capacity, this activity will eventually lead to bona-fide users being unable to connect since all the resources are being used by bogus connections.

This type of attack is easy to detect since it results in a sharp increase in network traffic, either as a whole, or from/to a host or subnet.

H. Domain Name Service (DNS)

This is an example of an attack in which an attacker gains control of a service used by hosts in the system. The DNS is used to translate host names into addresses. An attacker impersonating a remote name server could return fraudulent addresses to name queries, thus leading hosts to connect with machines (possibly controlled by the attacker) other than those expected.

The above are only a sample of the possible types of attacks which could occur over a network. They serve to illustrate the variety of possible attacks. It should be noted that none of the above attacks use any particular bugs in the implementation of services. If any such bugs exist, they will provide additional avenues for attack. In summary, any attack will consist of the three phases: preparation, execution, and effects; and it will either use a service or it will provide a service (to be used by the target).

V. 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 an access matrix, and so it is often referred to the Access Control Matrix. 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, and the current traffic pattern can be passed through that mask. Data passing through the mask should be brought to the attention to a security officer.

Designing patterns for all possible attacks is difficult at best and perhaps impossible. Therefore, the NSM generates a mask of normal traffic flow, and an inverse (as in a photographic inverse) is made of this normal traffic matrix. This new mask represents all traffic flow outside the normal traffic flow. The matrix for the current traffic flow is passed through this "abnormal" mask, and any data passing through is presented of the security officer.

Unfortunately the matrices for the network traffic are potentially enormous, especially if a larger dimensional matrix is considered. Even sparse matrix implementations contain a very large number of cells. Checking each cell against the mask may require more resources than available. 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.

Our groupings are based on the axes of the matrix. 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. If processing power is available, greater analysis may be conducted on groups which do not show abnormality to reduce chances that the probabilistic search presented an incorrect answer.

Other structured groupings may 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.

VI. DETAILS OF THE N.S.M.

This section examines the details of the NSM prototype. The NSM was built on a Sun-3/50 workstations 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.

A. 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, and they may 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 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 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 protocols.

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 link lists. This sparse matrix is shared with the matrix analyzer. In addition to communicating to 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 that a new communication has begun.

The link 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 connections nodes there are below it. The destination node contains the sum of all the packets which have passed between the source and 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 rules have yet to be turned into masks). 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, which is mentioned in part *C* of this section, 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 computer is marked as suspicious, its previous network activity can

be tracked.

B. 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 added.

Detecting unusual patterns, 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 shows most computers communicate almost exclusively with a very small subset of other computers (roughly three to five other hosts), and when these computers 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 to the traffic 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 the 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 made.

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.

C. NSM User Interface

The user interface for the NSM system 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.

(i) 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.

(ii) 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.

(iii) 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. (See Fig. 2.)

(iv) 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.

(v) Non-competition with 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.

VII. PERFORMANCE OF THE N.S.M.

The first analysis of the NSM's performance only included the rule base detection. Probabilistic detection testing will begin shortly; however, a computation of the actual data path space used has been calculated and will be described later. The NSM was tested for twenty days on the our Ethernet. Included in the test data were two simulated attacks. The attacker and the individual writing the rules did not discuss the attacks.

As mentioned previously, only a few simple rules are used; however, these rules have proven useful. The following are the rules used:

- If the total number of connections by a single service between two machines is greater than fifteen, then report.

- If a host communicates with more than fifteen other hosts using the telnet or login services, then report.

- If a connection is attempted to a nonexistent host, then report.

A total of 86 warnings, or approximately four warnings a day, were issued. Most of these warnings were generated by workstations running a large number of X window tools on a remote CPU and by mail connections from our central mail host. Some of the more interesting warnings are described below.

1) A number of hosts copied a large number of files via the File Transfer Protocol. File transfers of over three hundred files were observed more than once. Further investigation showed that people were backing up their files to other machines using ftp. At least one large ftp was to a host at Stanford containing a large number of public domain programs for personal computers.

2) Over three hundred fingers were initiated between two machines. The finger program is one of several services which provide information about the state of a system: who is currently using the system, which people have never logged in (and thus may have a default password), who has unread mail, etc. This information can be used to prepare for an actual attack, so the report caused some concern. Fortunately it was only one of our colleagues launching a simulated attack.

3) Over 150 mail messages were exchanged between two machines which normally do not exchange mail. Further analysis of the system files on one of the machines indicated that the mail was exchanged between only two users — one of them being root on one machine. Further investigations will continue.

4) On several occasions, over thirty login failures were recorded between a dial-up port and a host on our network but not under our control. Investigations are still continuing for these events as well.

5) Several warning were issued concerning a large number of connections made through an unknown service by several new HP workstations. The unknown service appears to be local to the new machines.

6) Finally, a computer game called "Empire" was initiated on a local host, and the game and Internet address were announced on the usenet network. Frequent warnings were issued concerning the number of hosts, often over twenty, which were communicating with our local machine.

Only one of the two simulated attacks was picked up by the rule base system. The other simulated attack involved moving massive amounts of data between two machines. Since the information was copied using only a single service connection and only between two legitimate hosts, the rules did not detect the attack. Assuming these two machines do not regularly exchange massive amounts of data, the probabilistic detection scheme would have probably detected this attack.

Although the probabilistic detection scheme (passing the traffic through masks), has not been tested yet, the actual data paths and the potential data paths have been measured. 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 num of network services by which the two hosts may communicate. The total number of possible data paths is then the number of host pairs possible multiplied by the number of services used.

A data path is considered to be used if at least one connection, on the average, is made on that path every two weeks. A calculation of the total number of data paths actually used on our Ethernet was 0.6% of total number of data paths possible. Therefore our sparse matrix represents only 0.6% of the potential matrix size, and the probability of a random network attack occuring on one of the normally used data paths is only 0.6%.

VIII. 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. A use of the network is considered suspicious if it is very dissimilar to previous uses (aka profiles) or 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 security officer control, the requests for data start at the top level and proceed downward. 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. We have used this model to generate trial attacks on the network and to determine the effectiveness of the NSM in detecting such attacks. 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. The attacks we generated all involve noticeable increases in activity at one of the three levels of our analysis hierarchy, and were easily detected by the NSM.

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 with synthetic attacks 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 [LAMP82].

REFERENCES

[DENN87] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engg.*, vol. SE-13, pp. 222-232, Feb. 1987.

[ESTR87] D. Estrin, "Controls for interoganization networks," *IEEE Transactions on Software Engineering*, vol. SE-13, Feb. 1987.

[GRMO84] F. T. Grampp and R. H. Morris, "Unix operating system security," *AT&T Bell Labs Technical Journal*, vol. 63, Oct. 1984.

[JSAC89] *IEEE Journal on Selected Areas in Communications*, special issue on Secure Communications, vol. SAC-7, May 1989.

[LAMP82] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, July 1982.

[LUNT88a] T. F. Lunt, et. al., "IDES: The enhanced prototype," Technical Report No. SRI-CSL-88-12, SRI International, Menlo Park, CA, Oct. 1988.

[LUNT88b] T. F. Lunt, "Automated audit trail analysis and intrusion detection: A survey," *Proc., 11th National Computer Security Conf.*, Baltimore, MD, Oct. 1988.

[MEBO76] R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed packet switching for local computer networks," *Communications of the ACM*, July 1976.

[NEBE89] R. M. Needham and S. M. Bellovin, "Security problems in the TCP/IP protocol suite," *Computer Communications Review*, vol. 19, April 1989.

[NESS87] D. M. Nessett, "Factors affecting distributed system security," *IEEE Transactions on Software Engineering*, vol. SE-13, Feb. 1987.

[NESH78] R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM*, vol. 21, Dec. 1978.

[NETW87] *IEEE Network Magazine*, special issue on Network Security, vol. 1, April 1987.

[NEWM87] 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.

[PARK83] D. B. Parker, *Fighting Computer Crime*, New York: Scribner's, 1983.

[RUHO87] L. Rutledge and L. Hoffman, "A survey of issues in computer network security," *Computers and Security*, vol. 5, pp. 296-308, 1986.

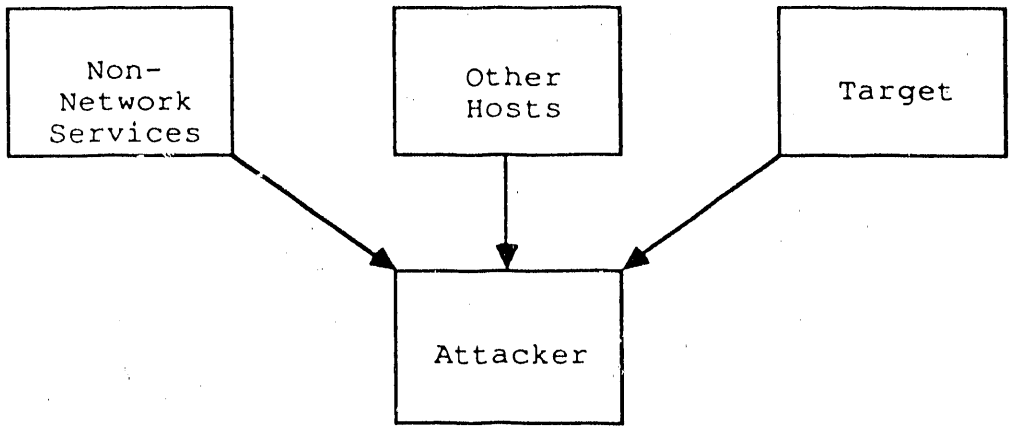
[STOL88] C. Stoll, "Stalking the wily hacker," *Communications of the ACM*, vol. 31, pp. 484-497, May 1988.

[TENS87] D. Tensa, "Typical weaknesses in operating systems software," *Information Age*, pp. 74-78, 1987.

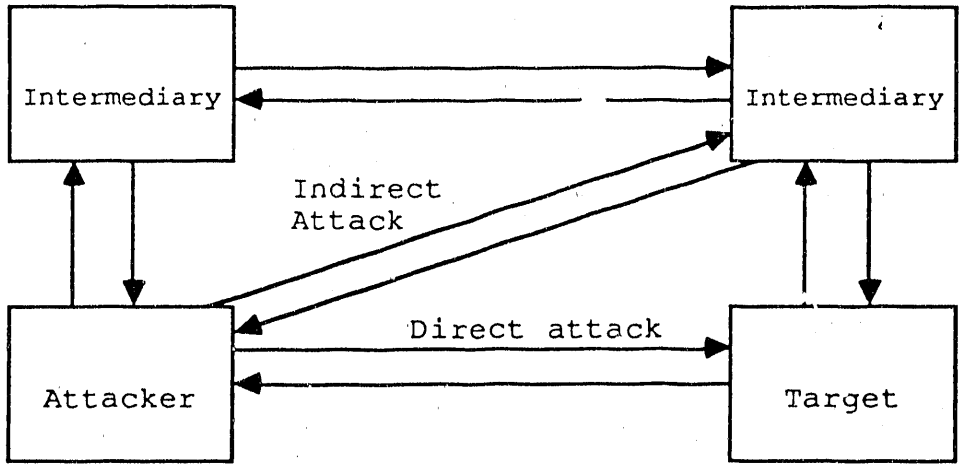
[VOKE85] V. Voydock and S. Kent, "Security in high-level network protocols," *IEEE Communications Magazine*, pp. 12-24, July 1985.

[WALK89] 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.*

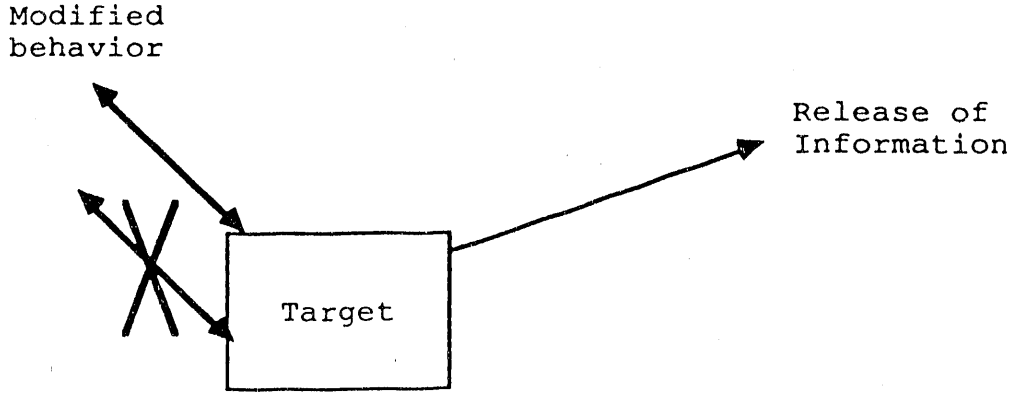
[WHIT87] R. A. Whitehurst, "Expert systems in intrusion detection: A case study," Computer Science Lab., SRI International, Menlo Park, CA, Nov. 1987.



Preparation Phase

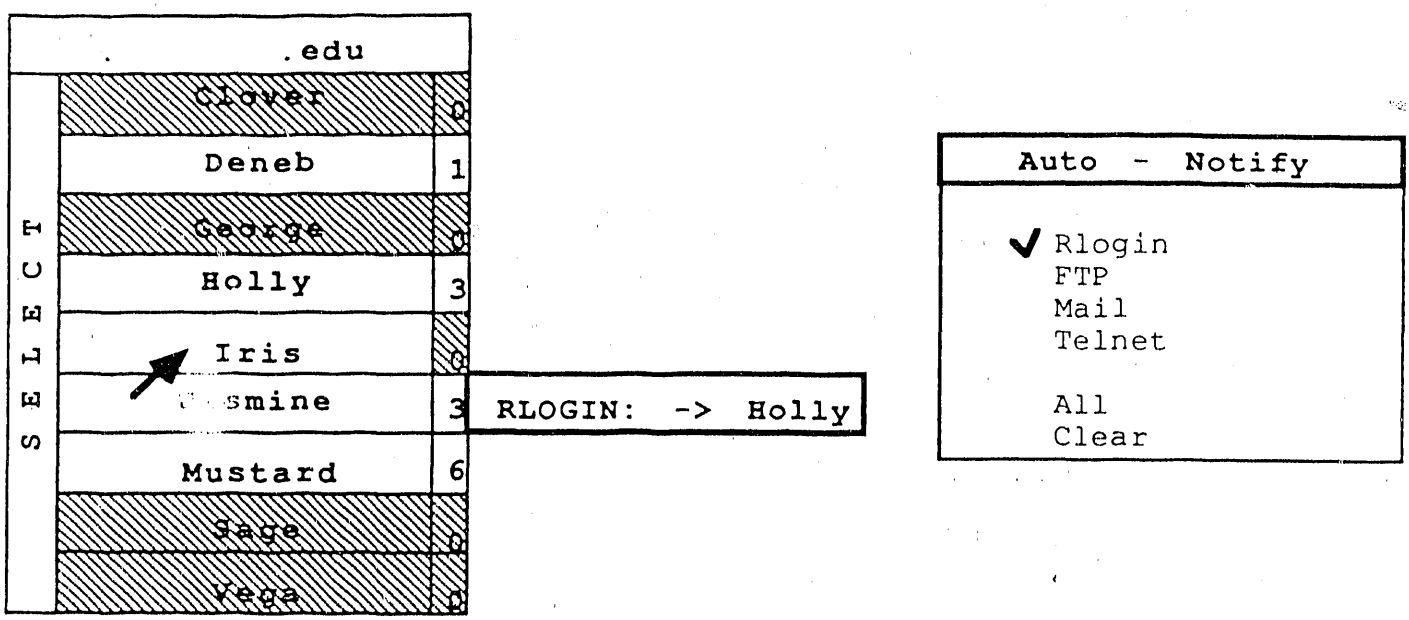


Attack Phase

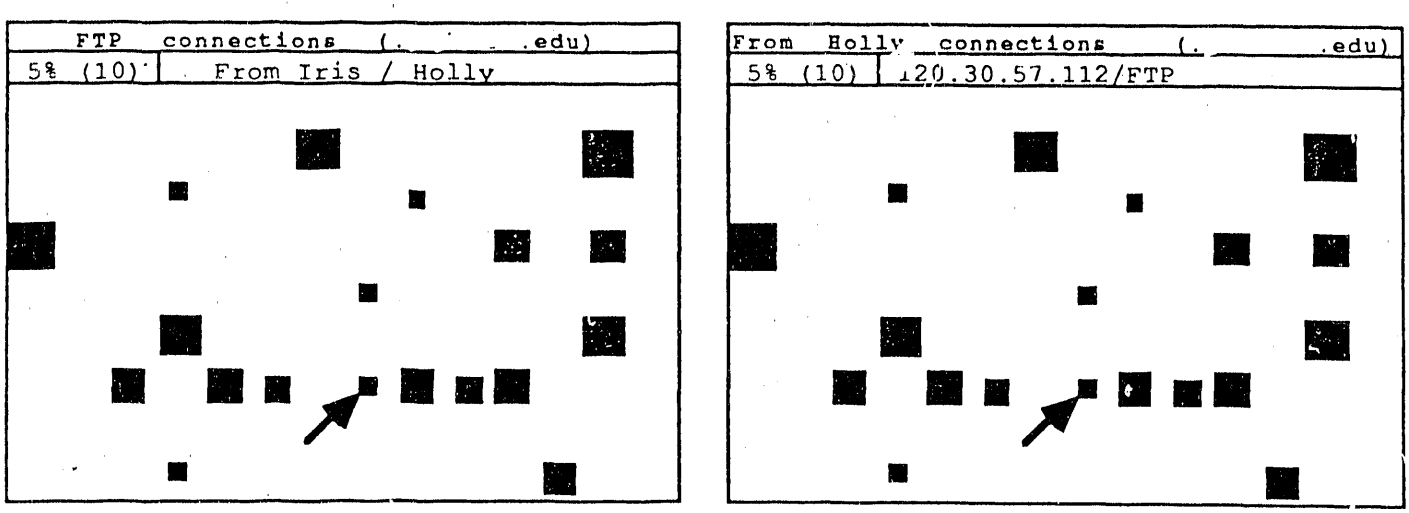


Post-Attack Phase

Fig. 1. Various phases of an attack.



Variable Display (0 constants)



GRID DISPLAY (1 constant)

Fig.2. Windows of the NSM interface.

CLOSE	Iris -> Holly
A: wood RLOGIN	% v <bs> l s <cr>NSMproject<tab>Mail<cr>% v <bs> l s <cr>
B: wood TELNET	m a n m a n <cr>Reformatting Page...W m a n m a n <cr>
C: RLOGIN	% f t p l e e k <cr> f t p l e e k <cr>
D: dias FTP	

CONNECTIONS FORM (2 constants)

Fig. 2. (contd.)

END

DATE FILMED

02 / 21 / 91

UCRL-CR-105095
B055743

Received by OSTI
FEB 0 6 1991

A NETWORK SECURITY MONITOR

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

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

November 1989

Lawrence Livermore National Laboratory

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract number W-7405-ENG-48.

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.