An Architecture for a Distributed Intrusion Detection System

James Brentano, Steven R. Snapp, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che-Lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha

Division of Computer Science
University of California
Davis, California 95616

Abstract

Intrusion detection is the problem of identifying unauthorized use, misuse, and abuse of computer systems. Outside attacks are not the only problem, the threat of authorized users misusing and abusing their privileges is an equally pressing concern. The proliferation of heterogeneous computer networks has additional implications for the intrusion detection problem. Namely, the increased connectivity of computer systems gives greater access to outsiders, and makes it easier for intruders to cover their tracks. Therefore, the goals of an intrusion detection system (IDS) are to use all available information in order to detect both attacks by external hackers and misuse by insiders. IDSs are based on the belief that an attacker's behavior will be noticeably different from that of a legitimate user. Given these goals and constraints, we have designed and implemented a Distributed Intrusion Detection System (DIDS) that combines distributed monitoring and data reduction with centralized data analysis to monitor a heterogeneous network of computers. DIDS consists of three different components: a single Host Monitor per host, a single LAN Monitor for each broadcast LAN segment, and a system Director. The DIDS Director is responsible for evaluation of the security state of the entire system of computers. It

This work supported in part by the USAF-AFCSC and the DOE through a grant from Lawrence Livermore National Laboratory

receives reports and information from each host and LAN Monitor and aggregates the information to make its decisions. The Host Monitor is primarily responsible for simple detection and reporting notable events. Similarly, the LAN Monitor watches all connections on its segment of the LAN, builds connections, and reports notable events. Initial system prototypes have provided quite favorable results, and further extensions are planned. This paper provides an in-depth discussion of the DIDS architecture and capabilities of the system.

1. Introduction

Intrusion detection is the problem of identifying individuals who are using a computer system without authorization and those who have legitimate access to the system but are abusing their privileges. These are often referred to as the *hacker* and *insider* threats respectively. Work is being done elsewhere on intrusion detection systems (IDSs) for a single host (e.g., Haystack, Wisdom & Sense, Midas) and even for several hosts connected by a network (e.g., IDES) [Snapp 1990]. Our own earlier work on the Network Security Monitor concentrated on monitoring a broadcast Local Area Network (LAN) [Heberlein 1990].

The proliferation of heterogeneous computer networks has serious implications for the intrusion detection problem. Foremost among these implications is the increased opportunity for illegitimate access that is provided by the network's connectivity. This problem is exacerbated when dial-up or internetwork access is allowed, as well as when unmonitored hosts are present. The use of distributed rather than centralized computing resources also implies reduced control over those resources. Moreover, multiple independent computers generate more audit data than a single computer performing the same amount of user work, and this audit data is dispersed among many systems.

This paper describes the Distributed Intrusion Detection System (DIDS) which monitors both the hosts connected to a network and the network itself. The information gathered by these distributed monitors is brought together and analyzed at a central location, thus providing the capability to correlate information from several different sources.

DIDS is designed to operate in an environment composed of C2 or higher rated computers connected by a broadcast LAN [Dept. of Defense 1985]. The use of C2 systems implies a consistency in the content of the system audit trails. This allows us to develop standard representations into which we can map audit data from UNIX, VMS or other C2 auditing systems. This is particularly important since we intend to deploy DIDS in a heterogenous environment. The C2 rating also guarantees, as part of the Trusted Computing Base (TCB), the security and integrity of the host's audit records. Although the hosts must comply with the C2 specification in order to be monitored directly, the network related activity of non-C2 hosts can be indirectly monitored via the LAN.

In the remainder of this paper we describe DIDS in detail. Section 2 motivates our work by describing some of the types of behavior which DIDS is intended to detect. In section 3 we present an overview of the distributed architecture of DIDS. In Section 4 we discuss the Network Identity (NID) and describe its use in distributed intrusion detection. Sections 5 and 6 deal with the Host and LAN monitors respectively. In section 7 we discuss some of the communication issues raised by the distributed architecture. Section 8 discusses the expert system and its relation to the NID. Section 9 consists of our concluding remarks.

2. Scenarios

There are a number of common attacks on networked computers which, for their detection, require information from multiple sources. One of the most common of these is the so-called *doorknob* attack. In a doorknob attack the goal is to discover, and gain access to, insufficiently protected systems. The intruder generally tries a few common account and password combinations on each of a number of computers. These simple attacks can be remarkably successful [Landreth 1985]. As a case in point, U.C. Davis' Network Security Monitor [Heberlein 1990], recently observed an attacker of this type gaining super-user access to a DOE computer which did not require a password for the super-user account. This particular intruder repeatedly tried to gain access to machines at a national laboratory using three account/password combinations. In this case the intruder used *telnet* to make the connection from a system at a university in another state. In cases like these, the intruder only tries a few logins on each machine, and tries them with different account names, which means that a traditional IDS may not notice the attack. Even if the behavior is recognized as an attack on the individual host, current IDSs are generally unable to correlate between reports from multiple hosts. Thus they cannot recognize the *doorknob* attack as such. Because DIDS aggregates and correlates data from multiple hosts, as well as the network, it is in a position to recognize the doorknob attack by detecting the pattern of repeated failed logins even though there may be too few on a single host to alert that host's monitor.

In another incident we recently observed, an intruder gained access to a computer using a "guest" account which didn't require a password. Once the attacker had access to the system he exhibited behavior which would have alerted most existing IDSs; for example, changing passwords, failed events, etc. In an incident such as this, DIDS would not only report the attack but would also identify the source of the attack. That is, while most IDSs would report that an incident involved user "guest" on the target machine, DIDS would also report that user "guest" was really, for example, user "smith" on the source machine, assuming that the source machine was in the monitored domain.

In addition to the specific scenarios outlined above, there are a number of general ways that an intruder can use the connectivity of the network to hide his trail and to enhance his effectiveness. Some of the attack configurations which have been hypothesized include *chain* and *parallel* attacks [Dias 1990]. DIDS combats these inherent vulnerabilities of the network by using the very same connectivity to help track and detect the intruder.

3. DIDS Architecture

The DIDS architecture combines distributed monitoring and data reduction with centralized data analysis (Fig. 1). This approach is unique among current IDSs. The components of DIDS are the *DIDS Director*, the *LAN Monitor* and the *Host Monitor*. There is a single Director, one LAN Monitor for each broadcast LAN segment, and one Host Monitor per host. The Host and LAN Monitors are primarily responsible for the collection of evidence while the DIDS Director is primarily responsible for its evaluation, Reports are sent asynchronously from the Host and LAN Monitors to the DIDS Director through the Communications Infrastructure. High level communication protocols between the components are based on the ISO Common Management Information Protocol (CMIP) recommendations, allowing for future inclusion of CMIP management tools as they become useful. The architecture also provides for bidirectional communication initiated by the DIDS Director. This communication consists primarily of requests by the Director for detailed information from the distributed monitors. The Director can also issue commands to the distributed monitors to modify their monitoring behavior. We perform some low level filtering and analysis on the distributed components and thereby minimize the use of network bandwidth in passing evidence to the Director. However, we maintain the benefits of centralized analysis, namely the ability to track users across the network, thereby providing a level of accountability not found in other systems.

The Host Monitor consists of a *Host Event Generator* (HEG) and a *Host Agent*. The HEG collects and analyzes audit records from the host's operating system. The audit records are scanned for *notable events*, which are transactions that are of interest independent of any other records. These include, but are not limited to, failed events, user authentications, and changes to the security state of the system. When notable events are found they are forwarded to the Director for further analysis. In the future, the HEG will also track user sessions and report anomalous behavior aggregated over time. The Host Agent handles the communication between the Host Monitor and the DIDS Director. It is responsible for dispatching reports to the DIDS Director and for handling requests and queries from the DIDS Director.

Like the Host Monitor, the LAN Monitor consists of a *LAN Event Generator* (LEG) and a *LAN Agent*. The LEG watches all of the traffic on its segment of the LAN. It audits host-to-host connections, services used, and volume of traffic. The LAN monitor reports to the Director network activity such as *rlogin* and *telnet* connections, the use of security related services, and significant changes in network load. The LAN Agent provides the same services to the LAN Monitor as the Host Agent does to the Host Monitor.

There are four components of the DIDS Director: the *Communications Manager*, the *Expert System*, a set of *Network Management & Incident Handling Tools* and the

User Interface. The Communications Manager is responsible for coordinating communication between the DIDS Director and the Host and LAN Monitors. The Expert System is responsible for evaluating the security state of the monitored network [Brentano 1991]. It receives the reports from each of the distributed monitors, correlates the data, and uses this information to reason about possible intrusions. Mechanisms are provided which allow the Expert System to be trained so that it reflects site specific conditions as well as to adapt to changes in those conditions. The User Interface is being designed to provide the system security officer with access to data regarding each of the hosts, the network, and the Expert System. It also provides access to the Network Management and Incident Handling Tools which are being developed separately.

4. The Network Identity

One of the most troublesome problems for intrusion detection in a networked environment is trying to track users and objects (e.g., files) across the network. For example, an intruder may use several different accounts on different machines during the course of an attack. This recognition can only be performed by correlating data from several independent sources, including the network itself. In a networked environment, an intruder will often make use of the interconnectivity of the computers to try to hide his true identity and location. The IDS must be able to determine the intruder's identity and location, not only to aid in the apprehension of the intruder, but also to help in detecting him in the first place. It may be that a single intruder uses multiple accounts to launch an attack, and that the behavior is suspicious only if one knows that all of the activity emanates from a single source. For example, it is not particularly noteworthy if a user inquires about who is using a particular computer (e.g., using the UNIX *who* command). However, it is extremely interesting if that same user inquires about who is using each of the computers on a LAN. It is even more indicative of an intrusion if the user subsequently logs in to one of the systems. Detecting this type of behavior requires being able to attribute multiple sessions, perhaps with different login names, to a single source. This problem is unique to the network environment and has not been dealt with before in this context. Our solution to the multiple user identity problem is to create a *network-user identity* (NID) for a user the first time he is seen anywhere in the monitored environment, and then to apply that identity to any further instances of the user. All evidence about the behavior of any instance of the user can then be applied to all other instances of the user on the network. In particular, we must be able to determine that smith@darth is the same person as jones@vader, if in fact they are.

Since the network identity problem involves the collection and evaluation of data from both the host and LAN Monitors, examining it is a useful way to understand the operation of DIDS. In the following subsections we examine each of the components of DIDS in the context of the creation and use of the NID.

5. The Host Monitor

The Host Monitor is currently installed on Sun SPARCstations running SunOS 4.0.x. with the Sun C2 security package installed [Sun 1990]. Through the C2 security package, the operating system produces audit records for virtually every transaction on the system. These transactions include file accesses, process executions, and logins. The general format of the Sun C2 audit record is:

record type
record event
time
real user ID
audit user ID
effective user ID
real group ID
process ID
error code
return value
label

The Host Monitor examines each audit record to determine if it should be forwarded to the Expert System for further evaluation. To do this the HEG creates a more abstract object called an *event*. The event includes the data provided by the original audit record plus two new fields: the *domain* and the *action*. The domain and action are abstractions which are used to minimize operating system dependencies at higher levels. Actions characterize the dynamic aspect of the audit records. Domains characterize the objects of the audit records. In most cases the objects are files or devices and their domain is

determined by the purpose of the object or its location in the file system. Since processes can also be objects of an audit record, they are also assigned to domains, in this case by their function.
The actions are:

session_start
session_end
read (a file or device)
write (a file or device)
execute (a process)
terminate (a process)
create (a file or (virtual) device)
delete (a file or (virtual) device)
move (rename a file or device)
change_rights
change_user_id

The domains are:

tagged
authentication
audit
network
system
sys_info
user_info
utility
owned
not_owned

The domains are prioritized so that an object is assigned to the first applicable domain. *Tagged* objects are ones which are thought a priori (e.g., by the security officer) to be particularly interesting in terms of detecting intrusions. Any file, device, or process can be tagged. *Authentication* objects are the processes and files which are used to provide access control on the system. Similarly, *audit* objects relate to the accounting and security auditing processes and files. *Network* objects are the processes and files not covered in the previous domains which relate to the use of the network. *System* objects are primarily those which are concerned with the execution of the operating system itself, again exclusive of those objects already assigned to previously considered domains. *Sys_info* and *user_info* objects respectively provide information about the system and about the users of the system, for example, *ps* and *who* in UNIX. The *utility* objects are the bulk of the programs run by the users;

for example, compilers and editors are in the utility domain. In general the execution of an object in the utility domain is not interesting but the creation or modification of one is. *Owned* objects are relative to the user. *Not_owned* objects are, by exclusion, every object not assigned to a previous domain. They are also relative to a user; thus files in the owned domain relative to SMITH are in the not_owned domain relative to JONES.

All possible transactions fall into one of a finite number of events formed by the cross product of the actions and the domains. Each event may also succeed or fail, so there are 220 distinct events. Note that no distinction is made between files, directories or devices, and that all of these are treated simply as objects. Not every action is applicable to every object; for example, the *terminate* action is applicable only to processes. The choice of these domains and actions is somewhat arbitrary, in that one could easily suggest both finer and coarser grained partitions. However, they capture most of the interesting behavior for intrusion detection and correspond reasonably well with what other researchers in this field have found to be of interest [Lunt 1990, Smaha 1990]. By mapping an infinite number of transactions to a finite number of events, we not only remove operating system dependencies but also restrict the number of permutations that the Expert System will have to deal with. The concept of the domain is one of the keys to detecting abuses. Using the domain allows us to make assertions about the nature of a user's behavior in a straightforward and systematic way. Although we lose some detail, that is more than made up for by the increase in portability, speed, simplicity and generality. The current system only considers events that are derived from a single audit record. In the future, it may be useful to consider events derived from a sequence of audit records using profiles or some other method. The syntax of an event reported by a Host Monitor is:

har(
Monitor ID,
Host ID,
Audit UID,
Real UID,
Effective UID,
Time,
Domain,
Action,
Transaction,
Object,
Parent Process,
PID,
Return Value,
Error Code).

Of the 110 possible events, only a subset are forwarded to the Expert System. For the creation and application of the NID it is the events which relate to the creation of user sessions are important. These include all the events with *session_start* actions, as well as ones with an *execute* action applied to the *network* domain. These latter events capture such transactions as executing the *rlogin, telnet, rsh,* and *rexec* UNIX programs. The HEG consults external tables to determine which events should be forwarded to the Expert System. These tables are built by hand. Because they relate to events rather than to the audit records themselves, the tables, and the modules of the HEG which use them, are portable across operating systems. The only portion of the HEG which is operating system dependent is the module which creates the events. This is an important consideration since we will be migrating the Host Monitor to other operating systems.

6. The LAN Monitor

The LAN Monitor is derived from UC Davis' Network Security Monitor [Heberlein 1990]. Since no native LAN audit trail exists, the LAN Monitor is responsible for building its own. The LAN Monitor sees every packet on its segment of the LAN and from these packets it is able to construct higher-level objects such as connections (logical circuits), and service requests using the TCP/IP and UDP/IP protocols. In particular, it audits host-to-host connections, services used, and volume of traffic per connection.

Like the Host Monitor, the LAN Monitor uses several kinds of simple analysis to identify significant events. They include the use of certain services as well as activity by certain classes of hosts, for example PCs without Host Monitors. The LAN Monitor also uses and maintains profiles of expected network behavior. The profiles consist of expected data paths (e.g., which systems are expected to establish communication paths to which other systems, and by which service) and service profiles (e.g., what is a typical *telent, mail, finger,* etc., expected to look like.) These profiles are updated periodically to keep them current.

The LAN Monitor also uses hueristics in an attempt to identify the likelihood that a particular connection represents intrusive behavior. These hueristics consider the capabilities of each of the network services, the level of authentication required for each of the services, the security level for each machine on the network, and signatures of past attacks. The abnormality of a connection is based on the probability of that particular connection occurring and the behavior of the connection itself. The LAN Monitor is also able, on request, to provide a more detailed examination of any connection, including capturing every character crossing an unencrypted network. This capability can be used to support a directed investigation of a particular subject or object. Like the Host Monitor, the LAN Monitor forwards relevant security information to the director through its LAN agent. The syntax is:

```
nar(
        Monitor ID,
        Source Host,
        Dest Host,
        Time,
        Service,
        Domain,
        Status).
```

7. The Communications Infrastructure

The DIDS Communications Infrastructure consists of two different types of processes. Each of the Host and LAN Monitors has its own Communications Agent, which is mainly responsible for the transfer of audit record information from the Monitor to the DIDS Director. These Agents will also serve to receive requests for more specific types of information from the Director, and then send the responses back to the Director. In order to facilitate the command and control of this information flow, the DIDS Director has a Communications Manager (CM). The CM accepts audit record, information, and control packets from each of the authorized Agents, formats them as required for the fact base, and then forwards them to the Expert System. When the Expert System needs more specific information from a Monitor for a directed investigation, it instructs the CM to send out a request packet containing a query to the specified Agent. The CM must keep a database of all Agents currently on-line in order to keep track of the authorized Agents and their locations.

There are several communication problems that must be dealt with due to the distributed nature of the system. Because we are dealing with multiple hosts, there will be some clock skew between the machines. Even a difference of a few seconds can be critical when attempting to correlate events from multiple sources using ordering and timing characteristics. To solve this problem, a single machine is chosen as a reference clock. In general, the reference computer will be the machine on which the DIDS Director resides. The CM corrects the time component of each audit record to normalize them with respect to the reference clock.

In order for the Expert System to consider each audit record only once, each record is assigned a unique integer number. The unique number is generated from two separate numbers located in the header information of each record packet. Each audit record is assigned a sequence number by the Host Monitor when it is generated; the range of sequence numbers is sufficient that they are unique within the system at any time. Each Host Monitor is also assigned a unique identification number when it signs on with the CM. Thus, the combination of id number and sequence number provides a unique record number for each record, as well as an ex post facto method of verifying the source of the audit record.

With numerous Agents continually sending audit records to the CM, the CM must always be ready to receive and buffer the incoming packets. Because of the short turn-around time for the CM to handle each packet, and the reliability of the socket connections, there is relatively little chance for the CM to develop an overflow condition.

The major communications problem that we have not yet addressed is the problem of authenticating the audit record packets as they arrive. That is, since any process with the CM socket address can send a packet, the CM must be able to verify that a record packet is valid. For the initial prototypes, we are assuming no authentication is necessary, however, this will not be acceptable for future production versions. There are several, well understood, ways to authenticate packets, including encryption, digital signatures, sequence numbers, passwords, etc. We are evaluating these methods based on system requirements, speed, and flexibility.

8. The Expert System

DIDS utilizes a rule based (or production) Expert System. The Expert System is written in Prolog and much of the form of the rule base comes from Prolog and the logic notation that Prolog implies. The Expert System uses rules derived from the hierarchical Intrusion Detection Model (IDM) [Brentano 1991]. The IDM describes the data abstractions used in inferring an attack on a network of computers. That is, it describes the transformation from the raw audit data provided by the operating systems and by a LAN monitor to high level hypotheses about intrusions and the overall security of the monitored environment. In abstracting and correlating data from the distributed sources the model builds a virtual machine which consists of all the connected hosts as well as the network itself. This unified view of the distributed system simplifies the recognition of intrusive behavior which spans individual hosts. The model is also applicable to the trivial network of a single computer.

The model is the basis of the rule base. It serves both as a description of the function of the rule base, and as a touchstone for the actual development of the rules. The IDM consists of 6 layers, each layer representing the result of a transformation performed on the data (see Table 1).

The objects of the first level of the model are the audit records provided by the host operating system, or by the LAN Monitor, or by a third party auditing package. The objects at this level are both syntactically and semantically dependent on the source. At this level all of the activity on the host, or LAN, is represented.

At the second level is the *Event* which is both syntactically and semantically independent of the source. Events have already been discussed in the context of the Host and LAN Monitor.

The third layer of the IDM creates a *subject.* This introduces a single identity for a user across many hosts on the network. It is the subject who is identified by the NID. Upper layers of the model treat the network user as a single entity, essentially ignoring his local identities on various hosts. Similarly, above this level, the collection of hosts on the LAN are generally treated as a single distributed system with little attention being paid to the individual hosts.

The fourth layer of the model introduces the "event in context". There are two kinds of context: temporal and spatial. As an example of temporal context, behavior which is unremarkable during working hours may be highly suspicious at other times [Lunt 1988]. The IDM therefore allows for the application of information about wall-clock time to the events it is considering. By wall-clock time is meant information about time of day, weekdays vs. weekends, holidays, as well as periods of expected increased activity. In addition to the consideration of external temporal context as described above, the Expert System uses time windows to correlate multiple events occurring in temporal proximity. This notion of temporal proximity implements the hueristic that a call to the UNIX *who* followed closely by a *login* or *logout* is more likely to be related to an intrusion than either of those events occuring alone. Spatial context implies the recognition of the importance of certain sources of events. That is, events related to a particular user, or from a particular host may be more likely to represent an intrusion than similar events from a different source. This allows us, for example, to distinguish between different degrees of security inherent in an un-monitored PC versus a monitoreedworkstation. The model also allows for the correlation of multiple events from the same user, or from the same source. In both of these cases multiple events are more noteworthy when they have a common element than when they don't.

The fifth layer of the model considers the *threats* to the network and the hosts connected to it. Events in context are combined to create threats. The threats are partitioned by the nature of the abuse and the nature of the target. In other words, what is the intruder doing, and what is he doing it to. Abuses are divided into *attacks*, *misuses*, and *suspicious acts*. Attacks represent abuses in which the state of the machine is changed. That is, the file system or process state is different after the attack than it was prior to the attack. Misuses represent out of policy behavior in which the state of the machine is not affected. Suspicious acts are events which, while not a violation of policy, are of interest to an IDS. For example, commands which provide information about the state of the system, or access online manuals are suspicious. The targets of abuse are characterized as being either *system* or *user* objects and also as being either *passive* or *active*. User objects are objects which are owned by non-privileged users and/or reside within a non-privileged user's directory hierarchy. System objects are the complement of user objects. Passive objects are files, including files containing program binaries. Active objects are essentially running processes. Thus there are 12 threats:

system_dynamic_attack,
system_static_attack,
user_dynamic_attack,
user_static_attack,
system_dynamic_misuse,
system_static_misuse,
user_dynamic_misuse,
user_static_misuse,
system_dynamic_suspicious_act,
system_static_suspicious_act,
user_dynamic_suspicious_act,
user_static_suspicious_act.

The argument can be made that these threats are too general, and that useful information about specific types of attacks is not included. We counter by claiming that a less general taxonomy must necessarily begin to include concepts which are specific to an operating system or an environment and that such a taxonomy is not appropriate given the intended application of this model.

At the highest level the model produces a numeric value between one and one hundred which represents the overall security of the network. The higher the number the less secure the network. This value is a function of all the threats for all the subjects on the system. Here again we treat the collection of hosts as a single distributed system. Although representing the security level of the system as a single value seems to imply some loss of information, it provides for a quick reference point to be used by the security officer. In fact, in the implementation, no information is lost since the Expert System maintains all the evidence used in calculating the security state in its internal database, and the security officer has access to that database.

In the context of the network user identification problem we are concerned primarily with the lowest three levels of the model: the Audit Data, the Event, and the Subject. The generation of the first two of these have already been discussed, the creation of the Subject is the focus of the following section.

The Expert System is responsible for applying the rules to the evidence provided by the monitors. In general, the rules do not change during the execution of the Expert System. What does change is a numerical value associated with each rule. This *Rule Value* (RV) represents our confidence that the rule is useful in detecting intrusions. Logically the rules have the form:

antecedent => consequence

where the antecedent is either a fact reported by one of the distributed monitors, or a consequence of some previously satisfied rule. The antecedent may also be a conjunction of these. The details of the manipulation of the rules and certainty values are discussed in [Brentano 1991]. The overall structure of the rule base is a tree rooted at the top. Thus, many facts at the bottom of the tree will lead to a few conclusions at the top of the tree.

The Expert System shell consists of approximately a hundred lines of Prolog source code. The shell is responsible for reading new facts reported by the distributed Monitors, attempting to apply the rules to the facts and hypotheses in the Prolog database, reporting suspected intrusions, and maintaining the various dynamic values associated with the rules and hypotheses. The syntax for rules is:

```
rule(123,100,(single,[A]),(B))).
```

where the first number is the rule number, the second number is the initial RV, A is the single antecedent, and B is the consequence. Conjunctive rules have the form:

```
rule(124,100,(and,[A,B,C]),(D))).
```

where A,B,C are antecedents and D is the consequence. Disjunctive rules are not allowed; that situation is dealt with by multiple rules with the same consequence.

8.1. Building the NID

The only legitimate ways to create an instance of a user are to login from a terminal, console or off-LAN source, to change user id in an existing instance, or to create additional instances (local or remote) from an existing instance. In each case, there is only one initial login from an external device. When this original login is detected a new unique *Network-user ID* (NID) is created. This NID is applied to every subsequent action generated by that user. When a user with a NID creates a new session (a new login) this new session is associated with his original NID. Thus the system maintains a single identity for each physical user. If an event, other than a login, is reported for a user who does not have a network identity, then that fact is reported as evidence of an attack.

We consider an instance of a user to be the tuple <*session_start, user_id, host_id, time*>. Thus each login creates a new instance of a user. In associating a NID with an instance of a user, the Expert System first tries to use an existing NID. If no NID can be found which applies to the instance, a new one is created. Trying to find an applicable existing NID consists of several steps. If a user changes identity (e.g., using UNIX's *su* command) on a host, the new instance is assigned the same NID as the previous identity. If a user performs a remote login from one host to another the new instance gets the same NID as the source instance. When no applicable NID is found the rule which creates a unique new NID is:

```
rule(111,1000,(and,[
/* login */
  hhar(_,Host1,AUID,_,_,Time1,_,session_start,_,_,_,_,_),
/* no NID for Host/ID pair */
  + (ih(net_user(NID,AUID,Host,_),_,_,_)),
/* exists some hhar */
  hhar(_,_,_,_,_,Time2,_,_,_,_,_,_,_),
/* at least 30 seconds older */
  Diff is Time2 - Time1,  Diff > 30,
/* generate new NID */
  newNID(X)
                                    ]),
/* create net_user */
  (net_user(X,AUID,Host1,Time1))).
```

The actual association of a NID with an instance is through the hypothesis *net_user*. For every event reported by the distributed monitors a new hypothesis is created. This new hypothesis, called a *subject*, is formed by the rule:

```
rule(110,100,(and,[
  har(Mon,Host,AUID,UID,EUID,Time,Dom,Act,
      Trans,Obj,Parent,PID,Ret,Err).
  net_user(NID,AUID,Host,_)
                              ]),
  subj(NID,Mon,Host,AUID,UID,EUID,Time,Dom,
       Act,Trans,Obj,Parent,PID,Ret,Err))).
```

The rule creates a subject, getting the NID from the net_user and the remaining fields from the HAR, if and only if both the user id and the host id match. It is through the use of the subject that the Expert System correlates a user's actions regardless of the login name or host ID.

There is at least one way to circumvent this implementation: consider hosts A and B which are on the monitored network, and hosts X and Y which are not. If an attacker connects from A to X, then from X to Y, and then from Y to B, the Expert System will not be able to recognize that it is the same user coming in to B as going out of A (see Fig. 2c). On the other hand a human expert would not be able to determine this either. There are also cases in which there is uncertainty about the identity of a user. When the connection is through an unmonitored host (see Fig. 2d) the Expert System cannot know for certain that the connection leaving the host is attributable to the connection entering the host. In this case the Expert System asserts that the two connections are attributed to the same user, but does so with a less certainty. Simultaneous connections also pose a problem (see Fig. 2e). If two or more connections originate from the same host at essentially the same time and the user names do not provide correspondence, the expert system cannot decide which user is which. In this case the expert system errs on the side of caution and attributes both connections to both users.

9. Conclusion

Our Distributed Intrusion Detections System is designed to address the shortcomings of current single host IDS. Intrusion detection systems designed for a network environment will become increasingly important as the number and size of LANs increase. We are currently completing the testing of our prototype system. This prototype has demonstrated the viability of our distributed architecture in solving the network-user identification problem. We have tested the system on a sub-network of Sun SPARCstations and it has correctly identified network users in a variety of scenarios. Work continues on the development and refinement of the rules, particularly those which can take advantage of knowledge about particular kinds of attacks. We are adding a fuzzy pattern matching component to the Host Monitor to detect patterns of audit records that resemble known attacks. In addition to the current Host Monitor which is designed to detect attacks on general purpose multi-user computers, we are designing monitors for application specific hosts such as file servers and gateways. In support of the ongoing development of DIDS we are researching the extension of our model to a hierarchical Wide Area Network environment. We are also beginning to investigate the theoretical limits of intrusion detection.

10. References

Brentano, James. *An Expert System for Detecting Attacks on Distributed Computer Systems.* Master's Thesis, University of California, Davis, Feb. 1991.

Department of Defense. *Trusted Computer Systems Evaluation Criteria.* National Computer Security Center, December 1985.

Dias, Gihan V., Karl N. Levitt and Biswanath Mukherjee. *Modeling Attacks on Computer Systems: Evaluating Vulnerabilities and Forming a Basis for Attack Detection.* Technical Report No. CSE-90-41, University of California, Davis, 1990.

Heberlein, L. T. et al. A Network Security Monitor. *Proceedings of IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1990.

Landreth, Bill. *Out of the Inner Circle, A Hacker's Guide to Computer Security.* Microsoft Press, Bellevue, 1985.

Lunt, Theresa. Automated Audit Trail Analysis and Intrusion Detection: A Survey. *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD., October 1988.

Sibert, W. Olin. Auditing in a Distributed System: SunOS MLS Audit Trails. *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD, October 1988.

Snapp, Steven R, James Brentano, et al.. Intrusion Detection Systems (IDS): A Survey of Existing Systems and A Proposed Distributed IDS Architecture. Technical Report No. CSE-91-7 ,University of California, Davis

Intrusion Detection Model

Level	Name	Explanation
6	Security State	overall network security level
5	Threat	definition of categories of abuse
4	Context	event placed in context
3	Subject	definition and disambiguation of network user
2	Event	OS independent representation of user action (finite number of these)
1	Data	Audit or OS provided data

Table 1

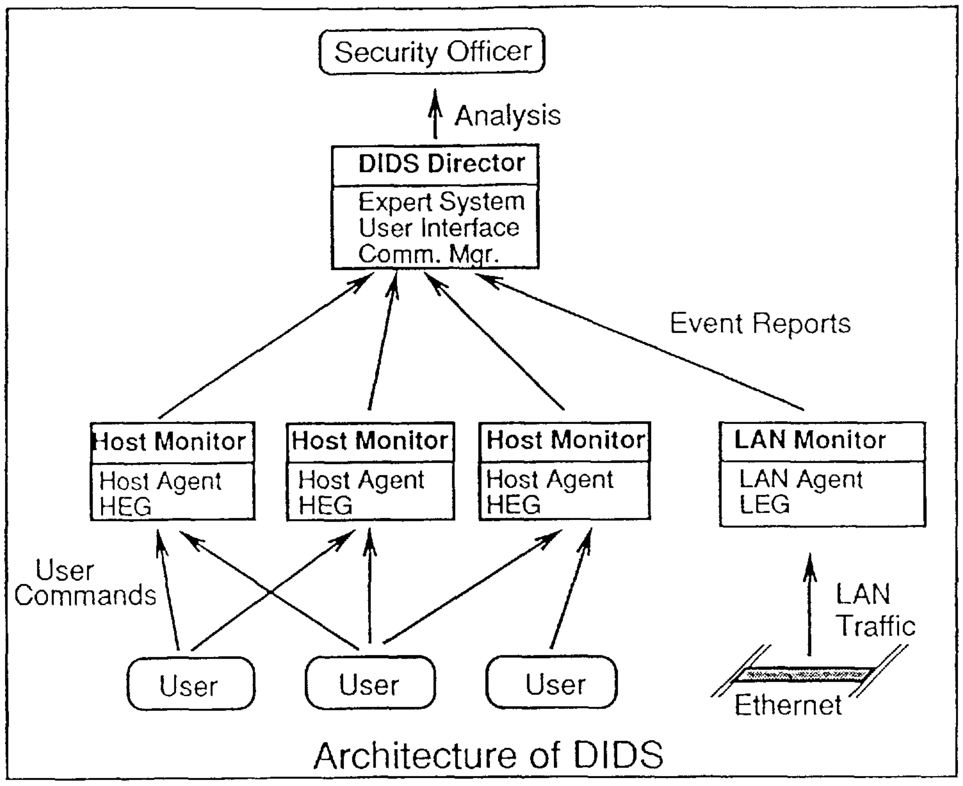

Security Officer
Analysis
DIDS Director
Expert System
User Interface
Comm. Mgr.
Event Reports
Host Monitor
Host Agent
HEG
Host Monitor
Host Agent
HEG
Host Monitor
Host Agent
HEG
LAN Monitor
LAN Agent
LEG
User Commands
LAN Traffic
User
User
User
Ethernet
Architecture of DIDS

Fig. 1

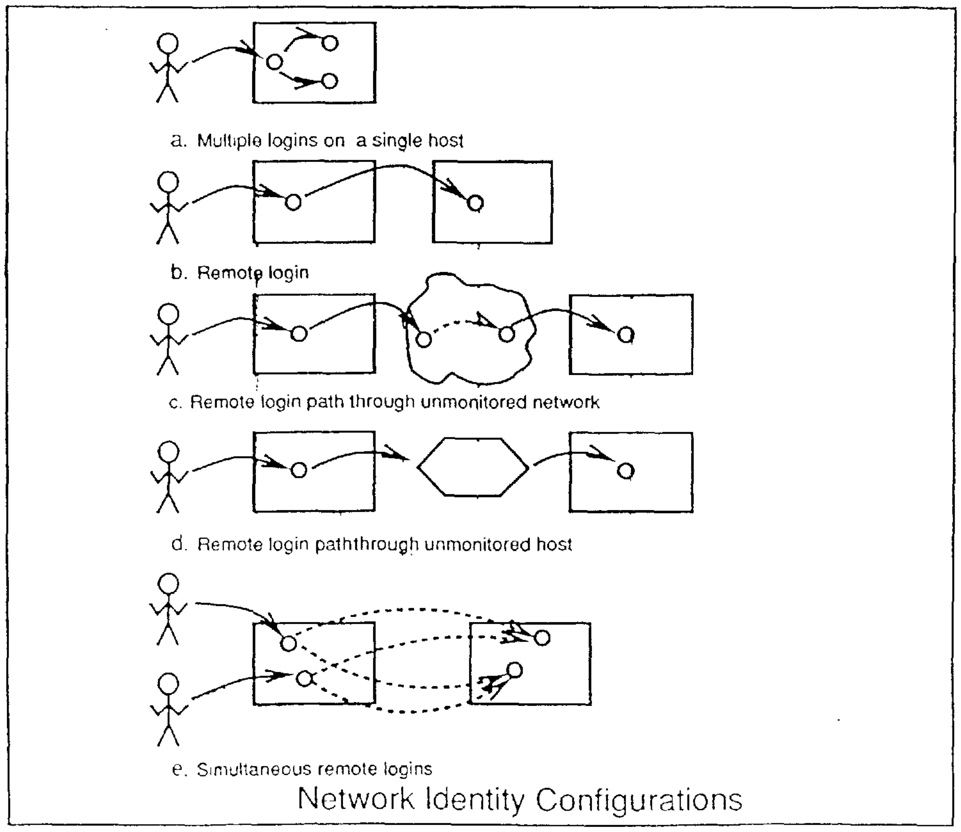
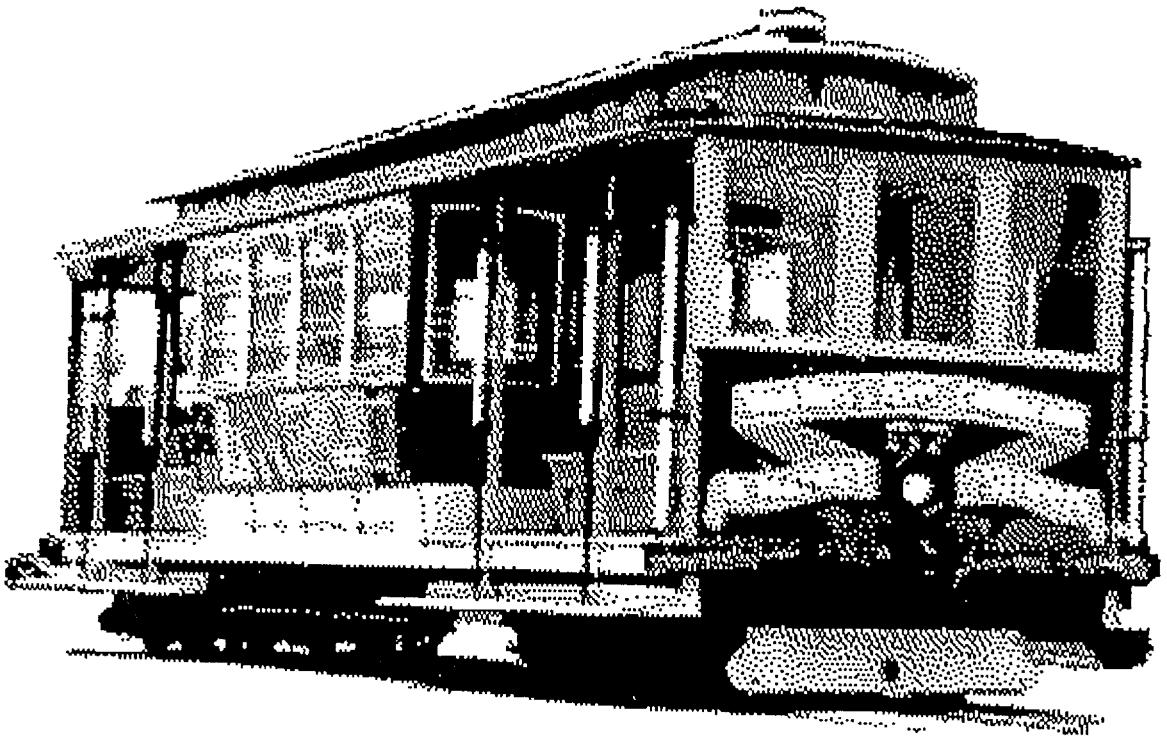

a. Multiple logins on a single host
b. Remote login
c. Remote login path through unmonitored network
d. Remote login paththrough unmonitored host
e. Simultaneous remote logins
Network Identity Configurations

Fig. 2

CØNF-9105126

14th Department of Energy Computer Security Group Conference

Proceedings



Concord, California *May 7 - 9, 1991*

Lawrence Livermore National Laboratory

"Computer Security - A Personal Commitment"

U.S. Department of Energy

Office of Administration and Human Resource Management
Office of Information Resources Management Policy, Plans and Oversight
and
Office of Security Affairs
Office of Safeguards and Security