DIDS (Distributed Intrusion Detection System) – Motivation, Architecture, and An Early Prototype

Steven R. Snapp[1], *James Brentano*[2], *Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che-Lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha*[1], *Tim Grance*[3], *Daniel M. Teal*[3], *and Doug Mansur*[4]

Computer Security Laboratory
Division of Computer Science
University of California, Davis
Davis, California 95616

ABSTRACT

Intrusion detection is the problem of identifying unauthorized use, misuse, and abuse of computer systems by both system insiders and external penetrators. The proliferation of heterogeneous computer networks provides 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 avoid detection. IDS's are based on the belief that an intruder's behavior will be noticeably different from that of a legitimate user. We are designing and implementing a prototype Distributed Intrusion Detection System (DIDS) that combines distributed monitoring and data reduction (through individual host and LAN monitors) with centralized data analysis (through the DIDS director) to monitor a heterogeneous network of computers. This approach is unique among current IDS's. A main problem considered in this paper is the Network-user Identification problem, which is concerned with tracking a user moving across the network, possibly with a new user-id on each computer. Initial system prototypes have provided quite favorable results on this problem and the detection of attacks on a network. This paper provides an overview of the motivation behind DIDS, the system architecture and capabilities, and a discussion of the early prototype.

1. Introduction

Intrusion detection is defined to be the problem of identifying individuals who are using a computer system without authorization (i.e., *crackers*) and those who have legitimate access to the system but are exceeding their privileges (i.e., the *insider threat*). Work is being done elsewhere on Intrusion Detection Systems (IDS's) for a single host [10[11]8] and for several hosts connected by a network [7[6]12]. Our own earlier work on the Network Security Monitor (NSM) concentrated on monitoring a broadcast Local Area Network (LAN) [3].

The proliferation of heterogeneous computer networks has serious implications for the intrusion detection problem. Foremost among these implications is the increased opportunity for unauthorized 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 (viz. hosts without audit trails) are present. The use of distributed rather than centralized computing resources also implies reduced control over those resources. Moreover, multiple independent computers are likely to generate more audit data than a single computer, and this audit data is dispersed among the various systems. Clearly, not all of the audit data can be forwarded to a single IDS for analysis; some analysis must be

[1] Haystack Laboratories, Inc., 8920 Business Park Dr, Suite 270, Austin, TX 78759

[2] Pacific Gas and Electric Company, 77 Beale St, Room 1871B, San Francisco, CA 94106

[3] United States Air Force Cryptologic Support Center, San Antonio, TX 78243

[4] Lawrence Livermore National Labs, Livermore, CA 94550

accomplished locally.

This paper describes a prototype Distributed Intrusion Detection System (DIDS) which generalizes the target environment in order to monitor multiple hosts connected via a network as well as the network itself. The DIDS components include the DIDS director, a single host monitor per host, and a single LAN monitor for each LAN segment of the monitored network. The information gathered by these distributed components is transported to, and analyzed at, a central location (viz. an expert system, which is a sub-component of the director), thus providing the capability to aggregate information from different sources. We can cope with any audit trail format as long as the events of interest are provided.

DIDS is designed to operate in a heterogeneous environment composed of C2 [1] or higher rated computers. The current target environment consists of several hosts connected by a broadcast LAN segment (presently an Ethernet, see Fig. 1). The use of C2-rated 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 any other system with C2 auditing capabilities. 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 specifications in order to be monitored directly, the network related activity of non-compliant hosts can be monitored via the LAN monitor. Since all attacks that utilize the network for system access will pass through the LAN segment, the LAN monitor will be able to monitor all of this traffic.

Section 2 motivates our work by describing the type of behavior which DIDS is intended to detect. In Section 3 we present an overview of the DIDS architecture. In Section 4 we formulate the concept of the network-user identification (NID), an identifier for a network-wide user, and describe its use in distributed intrusion detection. Sections 5 and 6 deal with the host and LAN monitors, respectively, while Section 7 discusses the expert system and its processing mechanisms based on the NID. Section 8 provides some concluding remarks.

2. Scenarios

The detection of certain attacks against a networked system of computers requires information from multiple sources. A simple example of such an attack is the so-called *doorknob* attack. In a doorknob attack the intruder's goal is to discover, and gain access to, insufficiently-protected hosts on a system. 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 [4]. As a case in point, UC Davis' NSM recently observed an attacker of this type gaining super-user access to an external computer which did not require a password for the super-user account. In this case, the intruder used *telnet* to make the connection from a university computer system, and then repeatedly tried to gain access to several different computers at the external site. In cases like these, the intruder tries only a few logins on each machine (usually with different account names), which means that an IDS on each host may not flag the attack. Even if the behavior is recognized as an attack on the individual host, current IDS's are generally unable to correlate reports from multiple hosts; thus they cannot recognize the *doorknob* attack as such. Because DIDS aggregates and correlates data from multiple hosts and 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, our NSM recently observed an intruder gaining access to a computer using a guest account which did not require a password. Once the attacker had access to the system, he exhibited behavior which would have alerted most existing IDS's (e.g., changing passwords and failed events). In an incident such as this, DIDS would not only report the attack, but may also be able to identify the source of the attack. That is, while most IDS's would report the occurrence of an incident involving 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. It may also be possible to go even further back and identify all of the different user accounts in the "chain" to find the initial launching point of the attack.

Another possible scenario is what we call *network browsing*. This occurs when a (network) user is looking through a number of files on several different computers within a short period of time. The browsing activity level on any single host may not be sufficiently high enough to raise any alarm by itself. However, the network-wide, aggregated browsing activity level may be high enough to raise suspicion on this user. Network browsing can be detected as follows. Each host monitor will report that a particular user is browsing on that system, even if the corresponding degree of browsing is small. The expert system can then aggregate such information from multiple hosts to determine that all of the browsing activity corresponds to the same network user. This scenario presents a key challenge for DIDS: the tradeoff between sending all audit records to the director versus missing attacks because

thresholds on each host are not exceeded.

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 [2]. DIDS combats these inherent vulnerabilities of the network by using the very same connectivity to help track and detect the intruder. Note that DIDS should be at least as effective as host-based IDS's (if we implement all of their functionality in the DIDS host monitor), and at least as effective as the stand-alone NSM.

3. DIDS Architecture

The DIDS architecture combines distributed monitoring and data reduction with centralized data analysis. This approach is unique among current IDS's. The components of DIDS are the *DIDS director*, a single *host monitor* per host. and a single *LAN monitor* for each broadcast LAN segment in the monitored network. DIDS can potentially handle hosts without monitors since the LAN monitor can report on the network activities of such hosts. The host and LAN monitors are primarily responsible for the collection of evidence of unauthorized or suspicious activity, while the DIDS director is primarily responsible for its evaluation. Reports are sent independently and asynchronously from the host and LAN monitors to the DIDS director through a communications infrastructure (Fig. 2). 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 between the DIDS director and any monitor in the configuration. This communication consists primarily of notable events and anomaly reports from the monitors. The director can also make requests for more detailed information from the distributed monitors via a "GET" directive, and issue commands to have the distributed monitors modify their monitoring capabilities via a "SET" directive. A large amount of low level filtering and some analysis is performed by the host monitor to minimize the use of network bandwidth in passing evidence to the director.

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, among others, failed events, user authentications, changes to the security state of the system, and any network access such as *rlogin* and *rsh*. These notable events are then sent to the director for further analysis. In enhancements under development, the HEG will also track user sessions and report anomalous behavior aggregated over time through user/group profiles and the integration of Haystack [10] into DIDS. The host agent handles all communications between the host monitor and the DIDS director.

Like the host monitor, the LAN monitor consists of a *LAN event generator* (LEG) and a *LAN agent*. The LEG is currently a subset of UC Davis' NSM [3]. Its main responsibility is to observe all of the traffic on its segment of the LAN to monitor host-to-host connections, services used, and volume of traffic. The LAN monitor reports on such network activity as *rlogin* and *telnet* connections, the use of security-related services, and changes in network traffic patterns.

The DIDS director consists of three major components that are all located on the same dedicated workstation. Because the components are logically independent processes, they could be distributed as well. The *communications manager* is responsible for the transfer of data between the director and each of the host and the LAN monitors. It accepts the notable event records from each of the host and LAN monitors and sends them to the *expert system*. On behalf of the expert system or user interface, it is also able to send requests to the host and LAN monitors for more information regarding a particular subject. The expert system is responsible for evaluating and reporting on the security state of the monitored system. It receives the reports from the host and the LAN monitors, and, based on these reports, it makes inferences about the security of each individual host, as well as the system as a whole. The expert system is a rule-based system with simple learning capabilities. The director's *user interface* allows the System Security Officer (SSO) interactive access to the entire system. The SSO is able to watch activities on each host, watch network traffic (by setting "wire-taps"), and request more specific types of information from the monitors.

We anticipate that a growing set of tools, including incident-handling tools and network-management tools, will be used in conjunction with the intrusion-detection functions of DIDS. This will give the SSO the ability to actively respond to attacks against the system in real-time. Incident-handling tools may consist of possible courses of action to take against an attacker, such as cutting off network access, a directed investigation of a particular user,

removal of system access, etc. Network-management tools that are able to perform network mapping would also be useful.

4. The Network-user Identification (NID)

One of the more interesting challenges for intrusion detection in a networked environment is to track users and objects (e.g., files) as they move across the network. For example, an intruder may use several different accounts on different machines during the course of an attack. Correlating data from several independent sources, including the network itself, can aid in recognizing this type of behavior and tracking an intruder to their source. In a networked environment, an intruder may often choose to employ the interconnectivity of the computers to hide his true identity and location. It may be that a single intruder uses multiple accounts to launch an attack, and that the behavior can be recognized as 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* or *finger* command). However, it may be indicative of an attack if a user inquires about who is using each of the computers on a LAN and then subsequently logs into one of the hosts. Detecting this type of behavior requires attributing multiple sessions, perhaps with different account 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 identification* (NID) the first time a user enters the monitored environment, and then to apply that NID to any further instances of the user. All evidence about the behavior of any instance of the user is then accountable to the single NID. In particular, we must be able to determine that "smith@host1" is the same user as "jones@host2", if in fact they are. Since the network-user identification problem involves the collection and evaluation of data from both the host and LAN monitors, examining it is a useful method 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 [9]. Through the C2 security package, the operating system produces audit records for virtually every transaction on the system. These transactions include file accesses, system calls, process executions, and logins. The contents of the Sun C2 audit record are: record type, record event, time, real user ID, audit user ID, effective user ID, real group ID, process ID, error code, return value, and label.

The host monitor (Fig. 3) examines each audit record to determine if it should be forwarded to the expert system for further evaluation. Certain critical audit records are always passed directly to the expert system (i.e., *notable events*); others are processed locally by the host monitor (i.e., *profiles* and attack *signatures*, which are sequences of noteworthy events which indicate the symptoms of attacks) and only summary reports are sent to the expert system. Thus, one of the design objectives is to push as much of the processing operations down to the low-level monitors as possible. In order to do this, the HEG creates a more abstract object called an *event*. The event includes any significant data provided by the original audit record plus two new fields: the *action* and the *domain*. The action and domain 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 characteristics 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, and change_user_id. The domains are: tagged, authentication, audit, network, system, sys_info, user_info, utility, owned, and 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 to be particularly interesting in terms of detecting intrusions. Any file, device, or process can be tagged (e.g., */etc/passwd*). *Authentication* objects are the processes and files which are used to provide access control on the system (e.g., the password file). 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 provide information about the system and about the users of the system, respectively. The *utility* objects are the bulk of the programs run by the users (e.g., compilers and editors). In general, the execution of an object in the utility domain is not interesting (except when the use is excessive), 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, and each event may also succeed or fail. 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 [510]. 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 details provided by the raw audit information, that is more than made up for by the increase in portability, speed, simplicity, and generality.

An event reported by a host monitor is called a host audit record (har). The record syntax 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 all the 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 or to a change in an account that 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, which are built by hand, to determine which events should be forwarded to the expert system. 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.

6. The LAN Monitor

The LAN monitor is currently a subset of UC Davis' Network Security Monitor [3]. The LAN monitor builds its own "LAN audit trail". The LAN monitor observes each and 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 or UDP/IP protocols. In particular, it audits host-to-host connections, services used, and volume of traffic per connection.

Similar to the host monitor, the LAN monitor uses several simple analysis techniques to identify significant events. The events include the use of certain services (e.g., *rlogin* and *telnet*) as well as activity by certain classes of hosts (e.g., a PC without a host monitor). 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 a typical *telnet, mail,* or *finger* is expected to look like).

The LAN monitor also uses heuristics in an attempt to identify the likelihood that a particular connection represents intrusive behavior. These heuristics 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. Upon request, the LAN monitor is also able to provide a more detailed examination of any connection, including capturing every character crossing the network (i.e., a wire-tap). 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.

An event reported by a LAN monitor is called a network audit record (nar). The record syntax is: nar(Monitor-ID, Source_Host, Dest_Host, Time, Service, Domain, Status).

The LAN monitor has several responsibilities with respect to the creation and use of the NID. The LAN monitor is responsible for detecting any connections related to *rlogin* and *telnet* sessions. Once these connections are detected, the LAN monitor can be used to verify the owner of a connection. The LAN monitor can also be used to help track tagged objects moving across the network. The SSO can also ask for a wire-tap on a certain network connection to monitor a particular user's behavior.

7. The Expert System

DIDS utilizes a rule-based (or production) expert system. The expert system is currently 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). The IDM describes the data abstractions used in inferring an attack on a network of computers. That is, it describes the transformation from the distributed raw audit data to high level hypotheses about intrusions and about 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 at the first level of the model are the audit records provided by the host operating system, 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, the *event* (which has already been discussed in the context of the host and LAN monitor) is both syntactically and semantically independent of the source standard format for events.

The third layer of the IDM creates a *subject*. This introduces a single identification for a user across many hosts on the network. It is the subject who is identified by the NID (see section 7.1). Upper layers of the model treat the network-user as a single entity, essentially ignoring the local identification on each host. 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 standard working hours may be highly suspicious during off hours [5]. The IDM, therefore, allows for the application of information about wall-clock time to the events it is considering. Wall-clock time refers to information about the time of day, weekdays versus weekends and holidays, as well as periods when an increase in activity is expected. In addition to the consideration of external temporal context, the expert system uses time windows to correlate events occurring in temporal proximity. This notion of temporal proximity implements the heuristic that a call to the UNIX *who* command followed closely by a *login* or *logout* is more likely to be related to an intrusion than either of those events occurring alone. Spatial context implies the relative importance of the source of events. That is, events related to a particular user, or events from a particular host, may be more likely to represent an intrusion than similar events from a different source. For instance, a user moving from a low-security machine to a high-security machine may be of greater concern than a user moving in the opposite direction. The model also allows for the correlation of multiple events from the same user or source. In both of these cases, multiple events are more noteworthy when they have a common element than when they do not.

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 may be suspicious. The targets of abuse are characterized as being either *system* objects or *user* objects and as being either *passive* or *active*. User objects 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 executable binaries, while active objects are essentially running processes.

At the highest level, the model produces a numeric value between one and 100 which represents the overall *security state* 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 a quick reference point for the SSO. In fact, in the current 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 SSO 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; thus, the creation of the subject is the focus of the following subsection.

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. These rule values are manipulated using a negative reinforcement training method which allows the expert system to continually lower the number of false attack reports. When a potential attack is reported by the expert system, the SSO determines the validity of the report and gives feedback to the expert system. If the report was deemed faulty, then the expert system lowers the RV's associated with the rules that were used to draw that conclusion. In addition to this directed training, which may lower some rule values, the system also automatically increases the RV's of all the rules on a regular basis. This recovery algorithm allows the system to adapt to changes in the environment as well as recover from faulty training.

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 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(n,r,(single,[A]),(C))).

where n is the rule number, r is the initial RV, A is the single antecedent, and C is the consequence. Conjunctive rules have the form:

rule(n,r,(and,[A_1,A_2,A_3]),(C))).

where A_1,A_2,A_3 are the antecedents and C is the consequence. Disjunctive rules are not allowed; that situation is dealt with by having multiple rules with the same consequence.

7.1. Building the NID

With respect to Unix, the only legitimate ways to create an instance of a user are for the user to login from a terminal, console, or off-LAN source, to change the 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 (system wide) from an external device. When this original login is detected, a new unique NID is created. This NID is applied to every subsequent action generated by that user. When a user with a NID creates a new login session, that new session is associated with his original NID. Thus the system maintains a single identification for each physical user.

We consider an instance of a user to be the 4-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 host, the new instance gets the same NID as the source instance. When no applicable NID is found, a new unique NID is created by the following rule:

```
rule(111,1000,[
  hhar(_,Host1,AUID,_,_,Time1,_,session_start,_,_,'local',_,_,_),  /* login */
  \+ (ih(net_user(NID,AUID,Host,_),_,_,_)),  /* no NID yet */
  newNID(X)  /* create new NID */
],
  (net_user(X,AUID,Host1,Time1))).  /* new net user */
```

The actual association of a NID with a user instance is through the hypothesis *net_user*. A new hypothesis is created for every event reported by the distributed monitors. 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 host audit record, 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 still some uncertainty involved with the network-user identification problem. If a user leaves the monitored domain and then comes back in with a different user-id, it is not possible to connect the two instances. Similarly, if a user passes through an unmonitored host, there is still uncertainty that any connection leaving the host is attributable to any connection entering the host. Multiple connections originating from the same host at approximately the same time also allow uncertainty if the user names do not provide any helpful information. The expert system can make a final decision with additional information from the host and LAN monitors that can (with high probability) disambiguate the connections.

8. Conclusion

Our Distributed Intrusion Detection System (DIDS) is being developed to address the shortcomings of current single host IDS's by generalizing the target environment to multiple hosts connected via a network (LAN). Most current IDS's do not consider the impact of the LAN structure when attempting to monitor user behavior for attacks against the system. Intrusion detection systems designed for a network environment will become increasingly important as the number and size of LAN's increase. Our 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 design, development, and refinement of rules, particularly those which can take advantage of knowledge about particular kinds of attacks. The initial prototype expert system has been written in Prolog, but it is currently being ported to CLIPS due to the latter's superior performance characteristics and easy integration with the C programming language. We are designing a signature analysis component for the host monitor to detect events and sequences of events that are known to be indicative of an attack, based on a specific context. In addition to the current host monitor, which is designed to detect attacks on general purpose multi-user computers, we intend to develop monitors for application specific hosts such as file servers and gateways. In support of the ongoing development of DIDS we are planning to extend our model to a hierarchical Wide Area Network environment.

Acknowledgments

The DIDS project is sponsored by the United States Air Force Cryptologic Support Center through a contract with the Lawrence Livermore National Labs.

References

1. Department of Defense, *Trusted Computer System Evaluation Criteria*, National Computer Security Center, DOD 5200.28-STD, Dec. 1985.
2. G.V. Dias, K.N. Levitt, and B. Mukherjee, ‘‘Modeling Attacks on Computer Systems: Evaluating Vulnerabilities and Forming a Basis for Attack Detection,’’ Technical Report CSE-90-41, University of California, Davis, Jul. 1990.
3. L.T. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber, ‘‘A Network Security Monitor,’’ *Proc. 1990 Symposium on Research in Security and Privacy*, pp. 296-304, Oakland, CA, May 1990.
4. B. Landreth, *Out of the Inner Circle, A Hacker's Guide to Compuer Security*, Microsoft Press, Bellevue, WA, 1985.
5. T. Lunt, ‘‘Automated Audit Trail Analysis and Intrusion Detection: A Survey,’’ *Proc. 11th National Computer Security Conference*, pp. 65-73, Baltimore, MD, Oct. 1988.
6. T.F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, H.S. Javitz, A. Valdes, and P.G. Neumann, ‘‘A Real-Time Intrusion-Detection Expert System (IDES),’’ Interim Progress Report, Project 6784, SRI International, May 1990.
7. T.F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P.G. Neumann, and C. Jalali, ‘‘IDES: A Progress Report,’’ *Proc. Sixth Annual Computer Security Applications Conference*, Tucson, AZ, Dec. 1990.
8. M.M. Sebring, E. Shellhouse, M.E. Hanna, and R.A. Whitehurst, ‘‘Expert Systems in Intrusion Detection: A Case Study,’’ *Proc. 11th National Computer Security Conference*, pp. 74-81, Oct. 1988.
9. W.O. Sibert, ‘‘Auditing in a Distributed System: SunOS MLS Audit Trails,’’ *Proc. 11th National Computer Security Conference*, Baltimore, MD, Oct. 1988.
10. S.E. Smaha, ‘‘Haystack: An Intrusion Detection System,’’ *Proc. IEEE Fourth Aerospace Computer Security Applications Conference*, Orlando, FL, Dec. 1988.
11. H.S. Vaccaro and G.E. Liepins, ‘‘Detection of Anomalous Computer Session Activity,’’ *Proc. 1989 Symposium on Research in Security and Privacy*, pp. 280-289, Oakland, CA, May 1989.
12. J.R. Winkler, ‘‘A Unix Prototype for Intrusion and Anomaly Detection in Secure Networks,’’ *Proc. 13th National Computer Security Conference*, pp. 115-124, Washington, D.C., Oct. 1990.

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. Intrusion Detection Model