

Frameworks for Network Intrusion Detection Systems: Wired and Wireless

J.RaviTeja

M.Tech, Student,

Department of Computer Science, K L University, Guntur.

V.Ramakrishna

M.Tech, Associate Professor,

Department of Computer Science, K L University, Guntur.

ABSTRACT

An intrusion can be defined as “any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource”. Intrusion prevention techniques, such as user authentication (e.g., using passwords or biometrics), avoiding programming errors, and information protection (e.g., encryption) have been used to protect computer systems as a first line of defense. The NetSTAT system was a network-based intrusion detection system. NetSTAT extended the state transition analysis technique (STAT) to network-based intrusion detection in order to represent attack scenarios in a networked environment. The NetSTAT approach models network attacks as state transition diagrams, where states and transitions are characterized in a networked environment. A NIDS can detect many types of events, from benign to malicious. Reconnaissance events alone are not dangerous, but can lead to dangerous attacks.

Keywords—Intrusion detection, Network tap, Network Security, Sensors, Spanning port

I. INTRODUCTION

As network-based computer systems play increasingly vital roles in modern society, they have become the targets of our enemies and criminals. Therefore, we need to find the best ways possible to protect our systems. The security of a computer system is compromised when an intrusion takes place. An intrusion can thus be defined as “any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource” [Heady et al., 1990]. Intrusion prevention techniques, such as user authentication (e.g., using passwords or biometrics), avoiding programming errors, and information protection (e.g., encryption) have been used to protect computer systems as a first line of defense. Intrusion prevention alone is not sufficient because as systems become ever more complex, there are always exploitable weaknesses in the systems due to design and programming errors, or various “socially engineered” penetration techniques. For example, after it was first reported many years ago, exploitable “buffer overflow” still exists in some recent system softwares due to programming errors. The policies that balance convenience versus strict control of a system and information access also make it impossible for an operational system to be completely secure. Intrusion detection is therefore needed as another wall to protect computer systems. The primary assumptions of intrusion detection are: user and program activities are observable, for example, via system auditing mechanisms; and more importantly, normal and intrusion activities have distinct behavior. Intrusion detection therefore includes these essential elements:

- Resources to be protected in a target system, for example, network services, user accounts, system kernels, etc.

- Models that characterize the “normal” or “legitimate” behavior of the activities involving these resources;
- Techniques that compare the observed activities with the established models. The activities that are not “normal” are flagged as “intrusive”. [1]

Currently, building effective IDS is an enormous knowledge engineering task. System builders rely on their intuition and experience to select the statistical measures for anomaly detection [Lunt, 1993]. Experts first analyze and categorize attack scenarios and system vulnerabilities, and hand-code the corresponding rules and patterns for misuse detection. Because of the manual and ad hoc nature of the development process, current IDSs have limited extensibility and adaptability.

II. Network Intrusion Detection Systems

A. Component Types

Two main component types comprise a NIDS: appliance and software only. A NIDS appliance is a piece of dedicated hardware: its only function is to be IDS. The operating system (OS), software, and the network interface cards (NIC) are included in the appliance. The second component type, software only, contains all the IDS software and sometimes the OS; however, the user provides the hardware. Software-only NIDSs are often less expensive than appliance-based NIDS because they do not provide the hardware; however, more configuration is required, and hardware compatibility issues may arise. With an IDS, the “system” component is vital to efficiency. Often a NIDS is not comprised of one device but of several physically separated components. Even in a less complicated NIDS, all components may be present but may be contained in one device. The NIDS is usually made of

components identified, but more specifically, the physical components usually include the sensor, management sever, database server, and console—

Sensor—the sensor or agent is the NIDS component that sees network traffic and can make decisions regarding whether the traffic is malicious. Multiple sensors are usually placed at specific points around a network, and the location of the sensors is important. Connections to the network could be at firewalls, switches, routers, or other places at which the network divides.

Management server—as the analyzer, a management server is a central location for all sensors to send their results. Management servers often connect to sensors *via* a management network; for security reasons, they often separate from the remainder of the network. The management server will make decisions based on what the sensor reports. It can also correlate information from several sensors and make decisions based on specific traffic in different locations on the network.

Database server—Database servers are the storage components of the NIDS. From these servers, events from sensors and correlated data from management servers can be logged. Databases are used because of their large storage space and performance qualities.

Console—as the user interface of the NIDS, the console is the portion of the NIDS at which the administrator can log into and configure the NIDS or to monitor its status. The console can be installed as either a local program on the administrator's computer or a secure Web application portal. Traffic between the components must be secure and should travel between each component unchanged and unviewed. Intercepted traffic could allow a hacker to change the way in which a network views an intrusion.

B. NIDS Sensor Placement

Because a sensor is the portion of the NIDS that views network traffic, its placement is important for detecting proper traffic. Figure 2 offers an example of how to place a NIDS sensor and other components. There are several ways to connect a NIDS sensor to the network—

Inline—an inline NIDS sensor is placed between two network devices, such as a router and a firewall. This means that all traffic between the two devices must travel through the sensor, guaranteeing that the sensor can analyze the traffic. An inline sensor of an IDS can be used to disallow traffic through the sensor that has been deemed malicious. Inline sensors are often placed between the secure side of the firewall and the remainder of the internal network so that it has less traffic to analyze.

Passive—a passive sensor analyzes traffic that has been copied from the network versus traffic that passes through it. The copied traffic can come from numerous places—

Spanning port—Switches often allow all traffic on the switch to be copied to one port, called a spanning port. During times of low network load, this is an easy way to view all traffic on a switch; however, as the load increases, the switch may not be able to copy all traffic. Also, if the switch deems the traffic malformed, it may not copy the

traffic at all; the malformed traffic that may be the type the NIDS sensor must analyze.

Network tap—a network tap copies traffic at the physical layer. Network taps are commonly used in fiber-optic cables in which the network tap is inline and copies the signal without lowering the amount of light to an unusable level. Because network taps connect directly to the media, problems with a network tap can disable an entire connection.[2]

C. Types of Events

A NIDS can detect many types of events, from benign to malicious. Reconnaissance events alone are not dangerous, but can lead to dangerous attacks. Reconnaissance events can originate at the TCP layer, such as a port scan. Running services have open ports to allow legitimate connections. During a port scan, an attacker tries to open connections on every port of a server to determine which services are running. Reconnaissance attacks also include opening connections of known applications, such as Web servers, to gather information about the server's OS and version. NIDS can also detect attacks at the network, transport, or application layers. These attacks include malicious code that could be used for denial of service (DoS) attacks and for theft of information. Lastly, NIDS can be used to detect less dangerous but nonetheless unwanted traffic, such as unexpected services (*i.e.*, backdoors) and policy violations.[3]

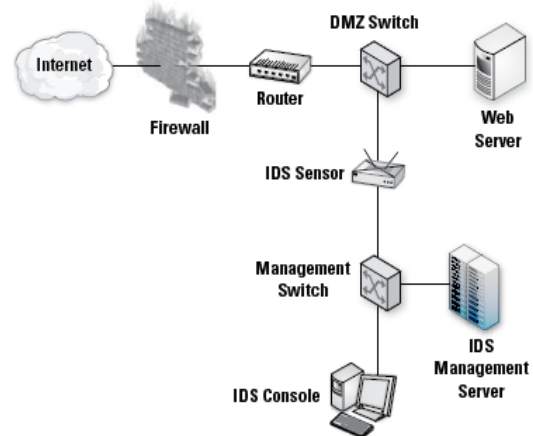


Figure 1: NIDS placement

D. Prevention

Although the detection portion of an IDS is the most complicated, the IDS goal is to make the network more secure, and the prevention portion of the IDS must accomplish that effort. After malicious or unwanted traffic is identified, using prevention techniques can stop it. When an IDS is placed in an inline configuration, all traffic must travel through an IDS sensor. When traffic is determined to be unwanted, the IDS does not forward the traffic to the remainder of the network. To be effective, however, this effort requires that all traffic pass through the sensor. When an IDS is not configured in an inline configuration, it must end the malicious session by sending a reset packet to the network. Sometimes the attack can happen before the

IDS can reset the connection. In addition, the action of ending connections works only on TCP, not on UDP or internet control message protocol (ICMP) connections. A more sophisticated approach to IPS is to reconfigure network devices (e.g., firewalls, switches, and routers) to react to the traffic. Virtual local area networks (VLAN) can be configured to quarantine traffic and limit its connections to other resources.[4]

III. THE NETSTAT SYSTEM

The NetSTAT system was a network-based intrusion detection system. NetSTAT extended the state transition analysis technique (STAT) to network-based intrusion detection in order to represent attack scenarios in a networked environment. However, unlike other network-based intrusion detection systems that monitored a single sub-network for patterns representing malicious activity, NetSTAT was oriented towards the detection of attacks in complex networks composed of several sub-networks. In this setting, the messages that are produced during an intrusion attempt may be recognized as malicious only in particular subparts of the network, depending on the network topology and service configuration. As a consequence, intrusions cannot be detected by a single component, and a distributed approach is needed. The NetSTAT approach models network attacks as state transition diagrams, where states and transitions are characterized in a networked environment. The network environment itself is described by using a formal model based on hypergraphs. The analysis of the attack scenarios and the network formal descriptions determines which events have to be monitored to detect an intrusion and where the monitors need to be placed. In addition, by characterizing in a formal way both the configuration and the state of a network it is possible to provide the components responsible for intrusion detection with all the information they need to perform their task autonomously with minimal interaction and traffic overhead. This can be achieved because network-based state transition diagrams contain references to the network topology and service configuration.

A. Architecture

NetSTAT is a distributed application composed of the following components: the network fact base, the state transition scenario database, a collection of general-purpose probes, and the analyzer. A high-level view of the NetSTAT architecture is given in Figure 2.

A.1 Network Fact Base

The network fact base component stores and manages the security-relevant information about a network. The fact base is a stand-alone application that is used by the Network Security Officer to construct, insert, and browse the data about the network being protected. It contains information about the network topology and the network services provided. The network topology is a description of the constituent components of the network and how they are connected. The network model underlying the NetSTAT tool uses interfaces, hosts, and links as primitive elements. A network is represented as a hyper graph on the set of

interfaces. In this model, interfaces are nodes while hosts and links are edges; that is, hosts and links are modeled as sets of interfaces. This is an original approach that has a number of advantages. Because the model is formal, it provides a well defined semantics and supports reasoning and automation.[5]

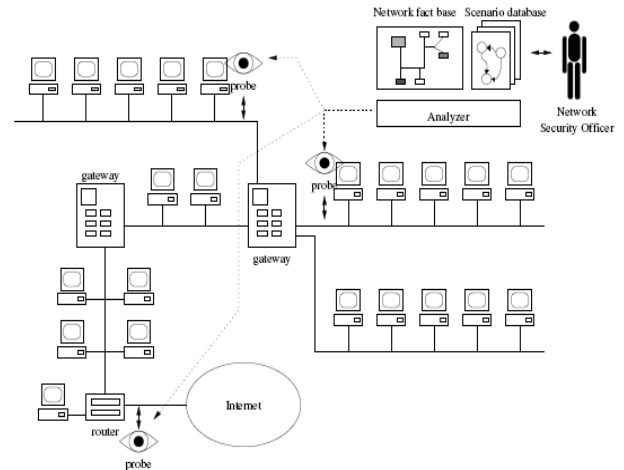


Figure 2: The NetSTAT architecture.

Another advantage is that this formalization allows one to model network links based on a shared medium (e.g., Ethernet) in a natural way, by representing the shared medium as a set containing all the interfaces that can access the communication bus. In this way, it is possible to precisely model the concept of network traffic eavesdropping, which is the basis for a number of network-related attacks. In addition, topological properties can be described in a simple way since hosts and links are treated uniformly as edges of the hypergraph. The network model is not limited to the description of the connection of elements. Each element of the model has some associated information. For example, hosts have several attributes that characterize the type of hardware and operating system software installed. The network services portion of the network fact base contains a description of the services provided by the hosts of a network. Examples of these services are the Network File System (NFS), the Network Information System (NIS), TELNET, FTP, "r" services, etc. The fact base contains a characterization of each service in terms of the network/transport protocol(s) used, the access model (e.g., request/reply), the type of authentication (e.g., address-based, password-based, token-based, or certificate-based), and the level of traffic protection (e.g., encrypted or not). In addition, the network fact base contains information about how services are deployed, that is, how services are instantiated and accessed over the network.

Figure 3 shows an example network. In the hypergraph describing the network, interfaces are represented as black dots, hosts are represented as circles around the corresponding interfaces, and links are represented as lines connecting the interfaces. The sample network is composed of five links, namely L1, L2, L3, L4, and L5, and twelve

hosts. Here in after, it is assumed that each interface has a single associated IP address, for example interface i7 is associated with IP address a7.

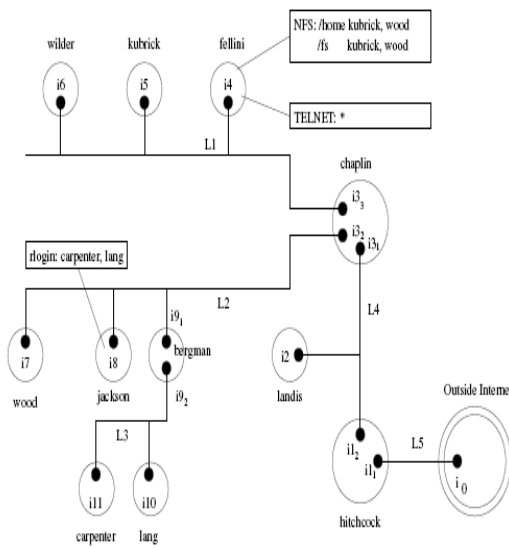


Figure 3: An example network.

The outside network is modeled as a composite host (the double circle in the figure) containing all the interfaces and corresponding addresses not in use elsewhere in the modeled network. As far as services are concerned, host Fellini is an NFS server exporting file systems /home and /fs to kubrick and wood. In addition, Fellini is a TELNET server for everybody. Host jackson exports an rlogin service to hosts carpenter and lang.[6]

A.2 State Transition Scenario Database

The state transition scenario database is the component that manages the set of state transition representations of the intrusion scenarios to be detected. The state transition analysis technique was originally developed to model host-based intrusions. It describes computer penetrations as sequences of actions that an attacker performs to compromise the security of a computer system. Attacks are (graphically) described by using state transition diagrams. States represent snapshots of a system’s volatile, semi-permanent, and permanent memory locations. A description of an attack has a “safe” starting state, zero or more intermediate states, and (at least) one “compromised” ending state. States are characterized by means of assertions, which are functions with zero or more arguments returning Boolean values. For NetSTAT the original STAT technique has been applied to computer networks, and the concepts of state, assertions, and signature actions have been characterized in a networked environment.

States and Assertions.

In network-based state transition analysis the state includes the currently active connections (for connection oriented services), the state of interactions (for connectionless services), and the values of the network tables (e.g., routing tables, DNS mappings, ARP caches, etc). For instance, both an open connection and a mounted file system are part of the state of the network. A pending

DNS request that has not yet been answered is also part of the state, such as the mapping between IP address 128.111.12.13 and the name hitchcock. For the application of state transition analysis to networks the original state transition analysis concept of assertion has been extended to include both static assertions and dynamic assertions. Static assertions are assertions on a network that can be verified by examining the network fact base; that is, by examining its topology and the current service configuration.

For example, the following assertion:

```
service s in server.services|
s.name == "www" and
s.application.name == "CERN httpd";
```

identifies a service s in the set of services provided by host server such that the name of the service is www and the application providing the service is the CERN http daemon1.

As another example, the following assertion:

```
Interface i in gateway.interfaces|
i.link.type == "Ethernet";
```

denotes an interface of a host, say gateway, that is connected to an Ethernet link.

These assertions are used to customize state transition representations for particular scenarios (e.g., a particular server and its clients). In practice, they are used to determine the amount of knowledge about the network fact base that each probe must be provided with during configuration procedures.

Dynamic assertions can be verified only by examining the current state of the network. One examples is NFS Mounted(filesystems, server, client), which returns true if the specified file system exported by server is currently mounted by client. Another example is ConnectionEstablished(addr1, port1, addr2, port2), which returns true if there is an established virtual circuit between the specified addresses and ports. These assertions are used to determine what relevant network state events should be monitored by a network probe.[7]

Transitions and Signature Actions.

In NetSTAT, signature actions are expressed by leveraging an event model. In this model, events are sequences of messages exchanged over a network. The basic event is the link-level message, or message for short. A link-level message is a string of bits that appears on a network link at a specified time. The message is exchanged between two directly-connected interfaces. For example the signature action:

```
Message m {i_x,i_y}|
m.length > 512;
```

represents a link-level message exchanged between interfaces i_x and i_y whose size is greater than 512 bytes.

Basic events can be abstracted or composed to represent higher-level actions. For example, IP datagrams that are transported from one interface to another in an IP network are modeled as sequences of link-level messages that represent the intermediate steps in the delivery process. Note that the only directly observable events are link-level messages appearing on specific links. Therefore, the IP

datagram “event” is observable by looking at the payload of one of the link-level messages used to deliver the datagram. For example, the signature action:

```
Message m in [IP Datagram d]{i_x, i_y}
m.dst != i_y;
```

represents a link-level message used during the delivery of an IP datagram such that the link-level destination address is not the final destination interface (i.e., the message is not the last one in the delivery process). Events representing single UDP datagrams or TCP segments are represented by specifying encapsulation in an IP datagram.

In the TCP virtual circuit case, application-level events are extracted by parsing the stream of bytes exchanged over the virtual circuit. The type of application event determines the protocol used to interpret the stream. For example, the following signature action:

```
[c.streamToServer [HTTPRequest r]]
r.method == "GET";
```

is an HTTP GET request that is transmitted over a TCP virtual circuit (defined somewhere else as c), through the stream directed to the server side.[8]

A.3 Probes

The probes are the active intrusion detection components. They monitor the network traffic in specific parts of the network, following the configuration they receive at startup from the analyzer, which is described in the following section. Probes are general-purpose intrusion detection systems that can be configured remotely and dynamically following any changes in the modeled attacks or in the implemented security policy. Each probe has the structure shown in Figure 4.

The filter module is responsible for filtering the network message stream. Its main task is to select those messages that contribute to signature actions or dynamic assertions used in a state transition scenario from among the huge number of messages transmitted over a network link. The filter module can be configured remotely by the analyzer. Its configuration can also be updated at run-time to reflect new attack scenarios, or changes in the network configuration.

The performance of the filter is of paramount importance, because it has strict real-time constraints for the process of selecting the events that have to be delivered to the inference engine. In the current prototype the filter is implemented using the BSD Packet Filter and a modified version of the tcp dump application

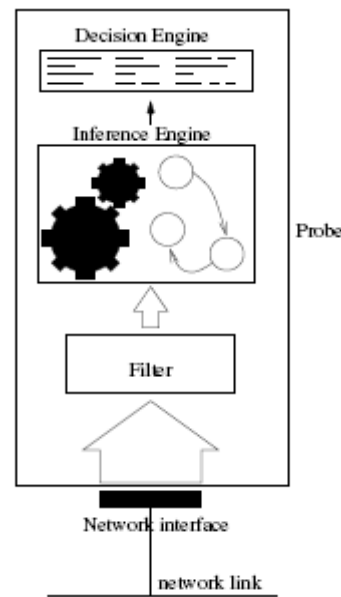


Figure 4: Probe architecture.

The inference engine is the actual intrusion detecting system. This module is initialized by the analyzer with a set of state transition information representing attack scenarios (or parts thereof). These attack scenarios are codified in a structure called the inference engine table. At any point during the probe execution, this table consists of snapshots of penetration scenario instances (instantiations), which are not yet completed. Each entry contains information about the history of the instantiation, such as the address and services involved, the time of the attack, and so on. On the basis of the current active attacks, the event stream provided by the filter is interpreted looking for further evidence of an occurring attack. Evolution of the inference engine state is monitored by the decision engine, which is responsible for taking actions based on the outcomes of the inference engine analysis. Some possible actions include informing the Network Security Officer of successful or failed intrusion attempts, alerting the Network Security Officer during the first phases of particularly critical scenarios, suggesting possible actions that can preempt a state transition leading to a compromised state, or playing an active role in protecting the network (e.g., by injecting modified datagrams that reset network connections.)

Probes are autonomous intrusion detection components. If a single probe is able to detect all the steps involved in an attack then the probe does not need to interact with any other probe or with the analyzer. Interaction is needed whenever different parts of an intrusion can be detected only by probes monitoring different subparts of the network. In this case, it is the analyzer’s task to decompose an intrusion scenario into sub-scenarios such that each can be detected by a single probe. The decision engine procedures associated with these scenarios are configured so that when part of a scenario is detected, an event is sent to the probes that are in charge of detecting the other parts of the overall attack. This simple form of forward chaining

allows one to detect attacks that involve different (possibly distant) sub-networks.[9]

A.4 Analyzer

The analyzer is used to analyze and instrument a network for the detection of a number of selected attacks. It takes as input the network fact base and a state transition scenario database and determines:

- which events have to be monitored; only the events that are relevant to the modeled intrusions must be detected;
- where the events need to be monitored;
- what information about the topology of the network is required to perform detection;
- what information must be maintained about the state of the network in order to be able to verify state assertions.

Thus, the analyzer component acts as a probe generator that customizes a number of general-purpose probes using an automated process based on a formal description of the network to be protected and of the attacks to be detected. This information takes the form of a set of probe configurations. Each probe configuration specifies the positioning of a probe, the set of events to be monitored, and a description of the intrusions that the probe should detect. These intrusion scenarios are customized for the particular sub-network the probe is monitoring, which focuses the scanning and reduces the overhead. The network fact base and the state transition scenario database components are used as internal modules for the selection and presentation of a particular network and a selected set of state transition scenarios. The analysis engine uses the data contained in the network fact base and the state transition scenario database to customize the selected attacks for the particular network under exam. For example, if one scenario describes an attack that exploits the trust relationship between a server and a client, that scenario will be customized for every client/server pair that satisfies the specified trust relationship3. [10]

Once the attack scenarios contained in the state transition scenario database have been customized over the given network, another module, called the configuration builder, translates the results of the analysis engine to produce the configurations to be sent to the different probes. Each configuration contains a filter configuration, a set of state transition information, and the corresponding decision tables to customize the probe's decision engine.

IV. Wireless

Wireless technologies have become so popular, and with the nature of wireless Communication blurring the borders between networks, special consideration is required. A wireless IDS is similar to an NIDS because the same types of network-based attacks can occur on wireless networks. However, because WLANs have other functionality and vulnerabilities, a WLAN IDS must monitor for network-based attacks as well as wireless specific attacks.

For WLANs, Wireless sensors may be standalone devices that are used to monitor all wireless traffic but without forwarding the traffic. Sensors may also be built

into wireless APs to monitor traffic as it connects to the wired network. The location of a WLAN sensor is important because its physical location affects what a sensor can monitor. A sensor should be able to monitor traffic from devices that can connect to the wireless network. (See Figure 5.) This could involve having several sensors that extend past the normal field of operations. WLAN devices operate on one channel at a time, but can choose from several. Consequently, a WLAN sensor can listen on only one channel at a time. Sensors can listen to either one channel or to several channels by changing them periodically, as one would change channels on a television. Several sensors may be used for listening to several channels at once.

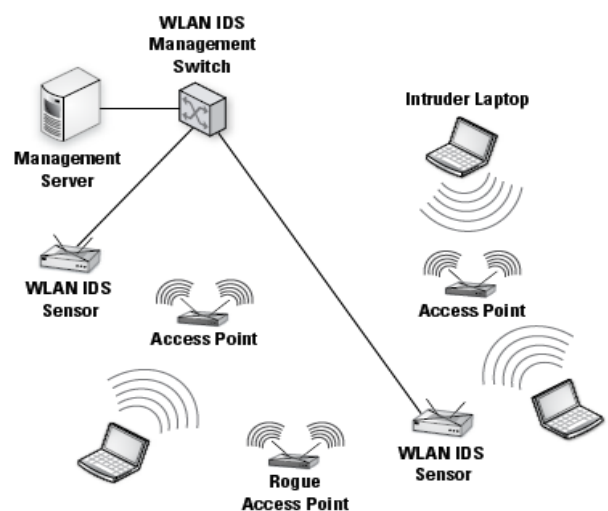


Figure 5: WLAN IDS placement

A. Components

A wireless IDS contains several components, such as sensors, management logging databases, and consoles, as does a NIDS. Wireless IDSs are unique in that they can be run centralized or decentralized. In centralized systems, the data is correlated at a central location and decisions and actions are made based on that data. In decentralized systems, decisions are made at the sensor.[11]

CONCLUSION

Intrusion detection and prevention systems are important parts of a well-rounded security infrastructure. IDSs are used in conjunction with other technologies (e.g., firewalls and routers), are part of procedures (e.g., log reviews), and help enforce policies. Each of the IDS technologies—NIDS, WLAN IDS and NETSTAT—are used together, correlating data from each device and making decisions based on what each type of IDS can monitor.

This “re-invention” of network intrusion detection techniques and approaches shows how intrusion detection (be it network-based, web-based, or host-based) is still an important research problem. As new attacks and new ways of compromising systems are introduced, both researchers

and practitioners will develop (or re-discover) techniques for the analysis of events that allow for the identification of the manifestation of malicious activity.

REFERENCES

1. Intrusion Detection Systems Information Assurance Tool Report, Sixth edition sep-25, 2009.
2. Allen, Julia; Christie, Alan; Fithen, William; McHugh, John; Pickel, Jed; Stoner, Ed. *State of the Practice of Intrusion Detection Technologies*. Pittsburg, PA: Carnegie Mellon Software Engineering Institute, January 2000
3. Base, Rebecca & Mell, Peter (2001). SP 800-31, *Intrusion Detection Systems*. Washington, DC: National Institute of Standards and Technology.
4. Giovanni Vigna *Network Intrusion Detection: Dead or Alive?* Department of Computer Science University of California, Santa Barbara
5. Kent, Karen & Mell, Peter (2006). SP 800-94, *Guide to Intrusion Detection and Prevention (IDP) Systems (DRAFT)*. Washington, DC: National Institute of Standards and Technology.
6. Kent, Karen & Warnock, Matthew (2004). *Intrusion Detection Tools Report, 4th Edition*. Herndon, VA: Information Assurance Technology Analysis Center (IATAC).
7. C. Berge. *Hypergraphs*. North-Holland, 1989. S. Eckmann, G. Vigna, and R. Kemmerer. *STATL: An Attack Language for State-based Intrusion Detection*. *Journal of Computer Security*, 10(1,2):71–104, 2002.
8. G. Vigna. A Topological Characterization of TCP/IP Security. In *Proceedings of the 12th International Symposium of Formal Methods Europe (FME '03)*, number 2805 in LNCS, pages 914–940, Pisa, Italy, September 2003. Springer-Verlag.
9. P. Porras. *STAT – A State Transition Analysis Tool for Intrusion Detection*. Master's thesis, Computer Science Department, University of California, Santa Barbara, June 1992.
10. Low, Christopher (2005). *Understanding wireless attacks & detection*. Bethesda, MD: The SANS Institute, Global Information Assurance Certification (GIAC) Security Essentials.
11. Thomas, Duncan. <http://compm067.paisley.ac.uk/notes/unit01.html>. ICT, Paisley University, 1999–2003.

Biographies

First Author:

Jammula Raviteja pursuing M.Tech in K.L.Univesity, Green Fields Vaddaswaram, Guntur. Area interest in research is Network Security.

Second Author:

V.Ramakrishna, M.Tech working as Associate Professor in K.L.Univesity, Green Fields Vaddaswaram, Guntur. Area of interest is software engineering and Network Security.