CYBER SECURITY

MODULE -4

SUBJECT CODE: 21AIM643

Prepared by : Prof. K.S. SHASHIKALA

MODULE-4	INTRUSION DETECTION	21AIM643.5 21AIM643.6	8 Hours
Host -Based Intrusion Detection – Network -Based Intrusion Detection – Distributed or Hybrid Intrusion Detection – Intrusion Detection Exchange Format – Honeypots – Example System Snort			

William Stallings, Lawrie Brown, "Computer Security Principles and Practice", Third Edition, Pearson Education, 2015.
(Chapter-8 –Module 4 ,Chapter 9- Module 5)

Security Intrusion:
A security event, or a combination of multiple security events, that constitutes a security incident in which an intruder gains, or attempts to gain, access to a system (or system resource) without having authorization to do so.

Intrusion Detection:

A security service that monitors and analyzes system events for the purpose of finding, and providing real-time or near real-time warning of, attempts to access system resources in an unauthorized manner

INTRUSION DETECTION SYSTEM(IDS)

Shashi KS

An IDS comprises three logical components:

• Sensors: Sensors are responsible for collecting data. The input for a sensor may be any part of a system that could contain evidence of an intrusion. Types of input to a sensor includes network packets, log files, and system call traces. Sensors collect and forward this information to the analyzer.

• Analyzers: Analyzers receive input from one or more sensors or from other analyzers. The analyzer is responsible for determining if an intrusion has occurred.

The output of this component is an indication that an intrusion has occurred. The output may include evidence supporting the conclusion that an intrusion occurred.

The analyzer may provide guidance about what actions to take as a result of the intrusion. The sensor inputs may also be stored for future analysis and review in a storage or database component.

• User interface: The user interface to an IDS enables a user to view output from the system or control the behavior of the system. In some systems, the user interface may equate to a manager, director, or console component.

IDSs are often classified based on the source and type of data analyzed, as:

• Host-based IDS (HIDS):

Monitors the characteristics of a single host and the events occurring within that host, such as process identifiers and the system calls they make, for evidence of suspicious activity.

• Network-based IDS (NIDS):

Monitors network traffic for particular network segments or devices and analyzes network, transport, and application protocols to identify suspicious activity.

• Distributed or hybrid IDS:

Combines information from a number of sensors, often both host and network-based, in a central analyzer that is able to better identify and respond to intrusion activity.

ANALYSIS APPROACHES

Shashi KS

IDSs typically use one of the following alternative approaches to analyze sensor data to detect intrusions:

1. Anomaly detection:

Involves the collection of data relating to the behavior of legitimate users over a period of time. Then current observed behavior is analyzed to determine with a high level of confidence whether this behavior is that of a legitimate user or alternatively that of an intruder.

2. Signature or Heuristic detection:

Uses a set of known malicious data patterns (signatures) or attack rules (heuristics) that are compared with current behavior to decide if is that of an intruder. It is also known as misuse detection. This approach can only identify known attacks for which it has patterns or rules.

Anomaly Detection

Shashi KS

The anomaly detection approach involves first developing a model of legitimate user behavior by collecting and processing sensor data from the normal operation of the monitored system in a training phase. This may occur at distinct times, or there may be a continuous process of monitoring and evolving the model over time. Once this model exists, current observed behavior is compared with the model in order to classify it as either legitimate or anomalous activity in a detection phase. A variety of classification approaches are used, which is broadly categorized as:

- Statistical: Analysis of the observed behavior using univariate, multivariate, or time-series models of observed metrics.
- Knowledge based:

Approaches use an expert system that classifies observed behavior according to a set of rules that model legitimate behavior.

- Machine-learning:

Approaches automatically determine a suitable classification model from the training data using data mining techniques.

A variety of machine-learning approaches have been tried, with varying success. These include:

• Bayesian networks:

Encode probabilistic relationships among observed metrics.

• Markov models:

Develop a model with sets of states, some possibly hidden, interconnected by transition probabilities.

• Neural networks: Simulate human brain operation with neurons and synapse between them, that classify observed data. • Fuzzy logic: Uses fuzzy set theory where reasoning is approximate, and can accommodate uncertainty.

• Genetic algorithms: Uses techniques inspired by evolutionary biology, including inheritance, mutation, selection and recombination, to develop classification rules.

• Clustering and outlier detection:

Group the observed data into clusters based on some similarity or distance measure, and then identify subsequent data as either belonging to a cluster or as an outlier.

Signature or Heuristic Detection

Shashi KS

Signature or heuristic techniques detect intrusion by observing events in the system and applying either a set of signature patterns to the data, or a set of rules that characterize the data, leading to a decision regarding whether the observed data indicates normal or anomalous behavior.

Signature approaches match a large collection of known patterns of malicious data against data stored on a system or in transit over a network. The signatures need to be large enough to minimize the false alarm rate, while still detecting a sufficiently large fraction of malicious data.

This approach is widely used in antivirus products, in network traffic scanning proxies, and in NIDS. The advantages of this approach include the relatively low cost in time and resource use, and its wide acceptance.

Disadvantages include the significant effort required to constantly identify and review new malware to create signatures able to identify it, and the inability to detect zero-day attacks for which no signatures exist. Rule-based heuristic identification involves the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses.

Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage.

Typically, the rules used in these systems are specific to the machine and operating system. The most fruitful approach to developing such rules is to analyze attack tools and scripts collected on the Internet. These rules can be supplemented with rules generated by knowledgeable security personnel. In this latter case, the normal procedure is to interview system administrators and security analysts to collect a suite of known penetration scenarios and key events that threaten the security of the target system.

The SNORT system, is an example of a rule-based NIDS. A large collection of rules exists for it to detect a wide variety of network attacks

HOST BASED INTRUSION DETECTION

Shashi KS

Host-based IDSs (HIDSs) add a specialized layer of security software to vulnerable or sensitive systems; such as database servers and administrative systems.
The HIDS monitors activity on the system in a variety of ways to detect suspicious behavior. In some cases, an IDS can halt an attack before any damage is done but its main purpose is to detect intrusions, log suspicious events, and send alerts.

The primary benefit of a HIDS is that it can detect both external and internal intrusions, something that is not possible either with network-based IDSs or firewalls.

Host-based IDSs can use either anomaly or signature and heuristic approaches to detect unauthorized behavior on the monitored host.

Data Sources and Sensors

Shashi KS

A fundamental component of intrusion detection is the sensor that collects data.

Some record of ongoing activity by users must be provided as input to the analysis component of the IDS. Common data sources include:

- System call traces:

A record of the sequence of systems calls by processes on a system, is widely acknowledged as the preferred data source for HIDS since the pioneering work of Forrest [CREE13]. While these work well on Unix and Linux systems, they are problematic on Windows systems due to the extensive use of DLLs that obscure which processes use specific system calls.

- Audit (log file) records :

Most modern operating systems include accounting software that collects information on user activity. The advantage of using this information is that no additional collection software is needed. The disadvantages are that the audit records may not contain the needed information or may not contain it in a convenient form, and that intruders may attempt to manipulate these records to hide their actions.

• File integrity checksums: A common approach to detecting intruder activity on a system is to periodically scan critical files for changes from the desired baseline, by comparing a current cryptographic checksums for these files, with a record of known good values. Disadvantages include the need to generate and protect the checksums using known good files, and the difficulty monitoring changing files. Tripwire is a well-known system using this approach.

• Registry access: An approach used on Windows systems is to monitor access to the registry, given the amount of information and access to it used by programs on these systems. However this source is very Windows specific, and has recorded limited success.

The sensor gathers data from the chosen source, filters the gathered data to remove any unwanted information and to standardize the information format, and forwards the result to the IDS analyzer, which may be local or remote

Anomaly HIDS

The majority of work on anomaly-based HIDS has been done on UNIX and Linux systems, given the ease of gathering suitable data for this work. While some earlier work used audit or accounting records, the majority is based on system call traces. System calls are the means by which programs access core kernel functions, providing a wide range of interactions with the low-level operating system functions. Hence they provide detailed information on process activity that can be used to classify it as normal or anomalous. Table 8.2 (a) lists the system calls used in current Ubuntu Linux systems as an example. This data is typically gathered using an OS hook, such as the BSM audit module. Most modern operating systems have highly reliable options for collecting this type of information. The system call traces are then analyzed by a suitable decision engine.

[CREE13] notes that the original work by Forrest et al introduced the Sequence Time-Delay Embedding (STIDE) algorithm, based on artificial immune system approaches, that compares observed sequences of system calls with sequences from the training phase to obtain a mismatch ratio that determines whether the sequence is normal or not. Later work has used other alternatives, such as Hidden Markov Models (HMM), Artificial Neural Networks (ANN), Support Vector Machines (SVM), or Extreme Learning Machines (ELM) to make this classification

these approaches all report providing reasonable intruder detection rates of 95-99% while having false positive rates of less than 5%, though on older test datasets.

Windows systems have traditionally not used anomaly based HIDS, as the wide usage of Dynamic Link Libraries (DLLs) as an intermediary between process requests for operating system functions and the actual system call interface has hindered the effective use of system call traces to classify process behavior. Some work was done using either audit log entries, or registry file updates as a data source, but neither approach was very successful. [CREE13] reports a new approach that uses traces of key DLL function calls as an alternative data source, with results comparable to that found with Linux system call trace HIDS

Table 8.2 (b) lists the key DLLs and executables monitored. Note that all of the distinct functions within these DLLs, numbering in their thousands, are monitored, forming the equivalent to the system call list presented in Table 8.2 (a). The adoption of this approach should lead to the development of more effective Windows HIDS, capable of detecting zero-day attacks, unlike the current generation of signature and heuristic Windows HIDS

Table 8.2 **Linux System Calls and Windows DLLs Monitored**

Shashi KS

(a) Ubuntu Linux System Calls

accept, access, acct, adjtime, aiocancel, aioread, aiowait, aiowrite, alarm, async_daemon, auditsys, bind, chdir, chmod, chown, chroot, close, connect, creat, dup, dup2, execv, execve, exit, exportfs, fchdir, fchmod, fchown, fchroot, fcntl, flock, fork, fpathconf, fstat, fstat, fstatfs, fsync, ftime, ftruncate, getdents, getdirentries, getdomainname, getdopt, getdtablesize, getfh, getgid, getgroups, gethostid, gethostname, getitimer, getmsg, getpagesize, getpeername, getpgrp, getpid, getpriority, getrlimit, getrusage, getsockname, getsockopt, gettimeofday, getuid, gtty, ioctl, kill, killpg, link, listen, lseek, lstat, madvise, mctl, mincore, mkdir, mknod, mmap, mount, mount, mprotect, mpxchan, msgsys, msync, munmap, nfs_mount, nfssvc, nice, open, pathconf, pause, pcfs_mount, phys, pipe, poll, profil, ptrace, putmsg, quota, quotactl, read, readlink, readv, reboot, recv, recvfrom, recvmsg, rename, resuba, rfssys, rmdir, sbreak, sbrk, select, semsys, send, sendmsg, sendto, setdomainname, setdopt, setgid, setgroups, sethostid, sethostname, setitimer, setpgid, setpgrp, setpgrp, setpriority, setquota, setregid, setreuid, setrlimit, setsid, setsockopt, settimeofday, setuid, shmsys, shutdown, sigblock, sigpause, sigpending, sigsetmask, sigstack, sigsys, sigvec, socket, socketaddr, socketpair, sstk, stat, stat, statfs, stime, stty, swapon, symlink, sync, sysconf, time, times, truncate, umask, umount, uname, unlink, unmount, ustat, utime, utimes, vadvise, vfork, vhangup, vlimit, vpixsys, vread, vtimes, vtrace, vwrite, wait, wait3, wait4, write, writev

(b) Key Windows DLLs and Executables

comctl32
kernel32
msvcpp
msvcrt
mswsock
ntdll
ntoskrnl
user32
ws2_32

OTHER APPROACHES

Shashi KS

Using system call traces provides arguably the richest information source for a HIDS, it does impose a moderate load on the monitored system to gather and classify this data.

The training phase for many of the decision engines requires very significant time and computational resources.

Hence others have trialed approaches based on audit (log) records. However these both have a lower detection rate than the system call trace approaches (80% reported), and are more susceptible to intruder manipulation.

A further alternative to examining current process behavior, is to look for changes to important files on the monitored host. This uses a cryptographic checksum to check for any changes from the known good baseline for the monitored files.

Typically all program binaries, scripts, and configuration files are monitored, either on each access, or on a periodic scan of the file system. The tripwire system is a widely used implementation of this approach, and is available for all major operating systems including Linux, Mac OSX and Windows.

Signature or Heuristic HIDS

Signature or heuristic based HIDS is widely used, particularly seen in anti-virus (A/V), more correctly viewed as anti-malware, products. These are very commonly used on Windows systems, and also incorporated into mail and web application proxies on firewalls and in network based IDSs. They use either a database of file signatures, which are patterns of data found in known malicious software, or heuristic rules that characterize known malicious behavior. These products are quite efficient at detecting known malware, however they are not capable of detecting zero-day attacks that do not correspond to the known signatures or heuristic rules. They are widely used, particularly on Windows systems, which continue to be targeted by intruders, a

Shashi KS

Distributed HIDS

Traditionally, work on host-based IDSs focused on single-system stand-alone operation. The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN or internetwork. Although it is possible to mount a defense by using stand-alone IDSs on each host, a more effective defense can be achieved by coordination and cooperation among IDSs across the network.

The major issues in the design of a distributed IDS are:

• A distributed IDS may need to deal with different sensor data formats. In a heterogeneous environment, different systems may use different sensors and approaches to gathering data for intrusion detection use.
• One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network. Thus, either raw sensor data or summary data must be transmitted across the network. Therefore, there is a requirement to assure the integrity and confidentiality of these data. Integrity is required to prevent an intruder from masking his or her activities by altering the transmitted audit information. Confidentiality is required because the transmitted audit information could be valuable.

- Either a centralized or decentralized architecture can be used. With a centralized architecture, there is a single central point of collection and analysis of all sensor data. This eases the task of correlating incoming reports but creates a potential bottleneck and single point of failure. With a decentralized architecture, there is more than one analysis center, but these must coordinate their activities and exchange information.

Architecture of a distributed IDS

Shashi KS

Figure 8.2 shows the overall architecture, of a distributed IDS, which consists of three main components:

1. Host agent module: An audit collection module operating as a background process on a monitored system. Its purpose is to collect data on securityrelated events on the host and transmit these to the central manager.

Figure 8.3 shows details of the agent module architecture.

2. LAN monitor agent module: Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager.

3. Central manager module: Receives reports from LAN monitor and host agents and processes and correlates these reports to detect intrusion. The scheme is designed to be independent of any operating system or system auditing implementation.

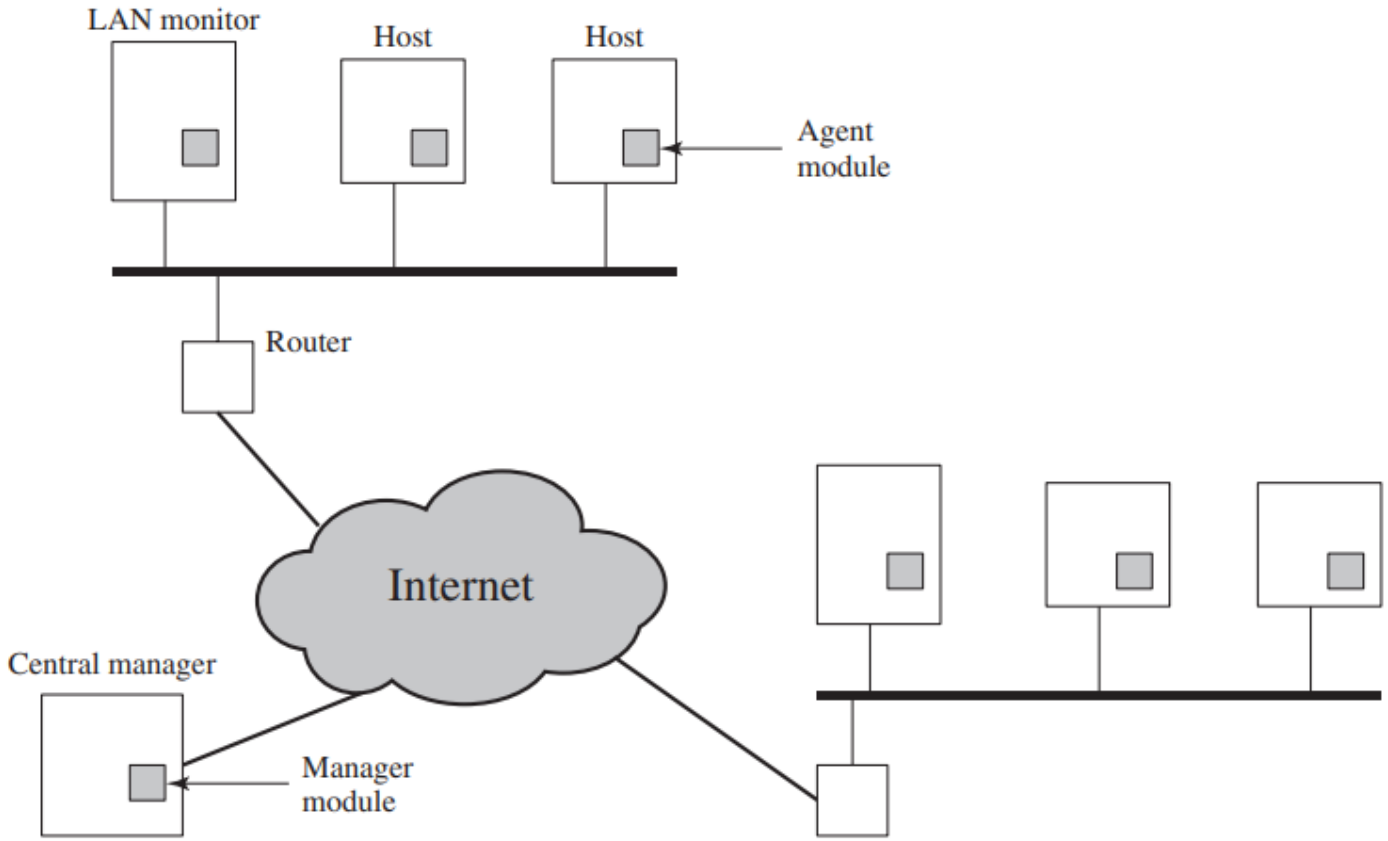

Figure 8.2 **Architecture for Distributed Intrusion Detection**

AGENT ARCHITECTURE

Shashi KS

Figure 8.3 shows the general approach that is taken. The agent captures each audit record produced by the native audit collection system. A filter is applied that retains only those records that are of security interest. These records are then reformatted into a standardized format referred to as the host audit record (HAR).

Next, a template-driven logic module analyzes the records for suspicious activity. At the lowest level, the agent scans for notable events that are of interest independent of any past events. Examples include failed files, accessing system files, and changing a file's access control. At the next higher level, the agent looks for sequences of events, such as known attack patterns (signatures).

Finally, the agent looks for anomalous behavior of an individual user based on a historical profile of that user, such as number of programs executed, number of files accessed, and the like. When suspicious activity is detected, an alert is sent to the central manager.

The central manager includes an expert system that can draw inferences from received data. The manager may also query individual systems for copies of HARs to correlate with those from other agents.

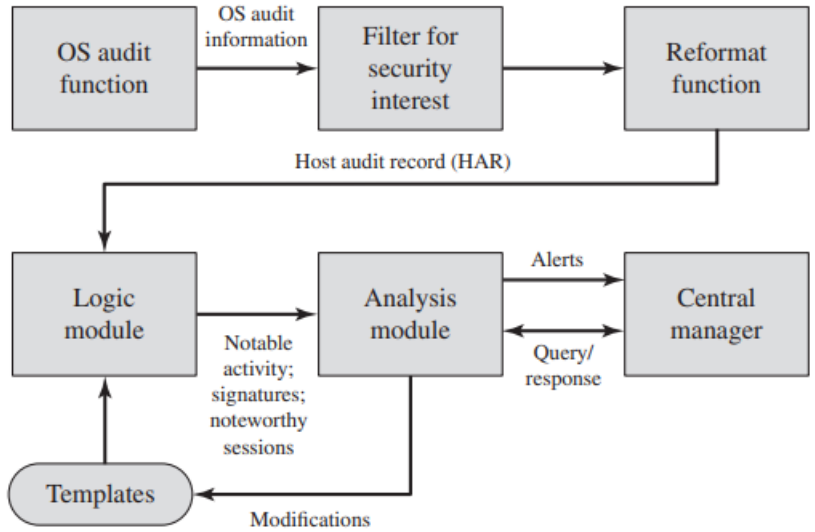

Figure 8.3 **Agent Architecture**

The LAN monitor agent also supplies information to the central manager. The LAN monitor agent audits host-host connections, services used, and volume of traffic. It searches for significant events, such as sudden changes in network load, the use of security-related services, and suspicious network activities.

Network-based IDS (NIDS)

Shashi KS

A network-based IDS (NIDS) monitors traffic at selected points on a network or interconnected set of networks. The NIDS examines the traffic packet by packet in real time, or close to real time, to attempt to detect intrusion patterns. The NIDS may examine network-, transport-, and/or application-level protocol activity.

Note the contrast with a host-based IDS; a NIDS examines packet traffic directed toward potentially vulnerable computer systems on a network. A host-based system examines user and software activity on a host

NIDS are typically included in the perimeter security infrastructure of an organization, either incorporated in, or associated with, the firewall.

They typically focus on monitoring for external intrusion attempts, by analyzing both traffic patterns and traffic content for malicious activity. With the increasing use of encryption though, NIDS have lost access to significant content, hindering their ability to function well. Thus while they have an important role to play, they can only form part of the solution.

A typical NIDS facility includes a number of sensors to monitor packet traffic, one or more servers for NIDS management functions, and one or more management consoles for the human interface. The analysis of traffic patterns to detect intrusions may be done at the sensor, at the management server, or some combination of the two.

Shashi KS

Types of Network Sensors

Sensors can be deployed in one of two modes: inline and passive.

An inline sensor is inserted into a network segment so that the traffic that it is monitoring must pass through the sensor. One way to achieve an inline sensor is to combine NIDS sensor logic with another network device, such as a firewall or a LAN switch. This approach has the advantage that no additional separate hardware devices are needed; all that is required is NIDS sensor software. An alternative is a stand-alone inline NIDS sensor. The primary motivation for the use of inline sensors is to enable them to block an attack when one is detected. In this case the device is performing both intrusion detection and intrusion prevention functions.

More commonly, passive sensors are used. A passive sensor monitors a copy of network traffic; the actual traffic does not pass through the device. From the point of view of traffic flow, the passive sensor is more efficient than the inline sensor, because it does not add an extra handling step that contributes to packet delay.

Typical passive sensor configuration

Shashi KS

Figure 8.4 illustrates a typical passive sensor configuration. The sensor connects to the network transmission medium, such as a fiber optic cable, by a direct physical tap. The tap provides the sensor with a copy of all network traffic being carried by the medium.

The network interface card (NIC) for this tap usually does not have an IP address configured for it. All traffic into this NIC is simply collected with no protocol interaction with the network. The sensor has a second NIC that connects to the network with an IP address and enables the sensor to communicate with a NIDS management server. Another distinction is whether the sensor is monitoring a wired or wireless network.

A wireless network sensor may either be inline, incorporated into a wireless access point (AP), or a passive wireless traffic monitor. Only these sensors can gather and analyze wireless protocol traffic, and hence detect attacks against those protocols. Such attacks include wireless denial-of-service, session hijack, or AP impersonation. A NIDS focussed exclusively on a wireless network is known as a Wireless IDS (WIDS). Alternatively wireless sensors may be a component of a more general NIDS gathering data from both wired and wireless network traffic, or even of a distributed IDS combining host and network sensor data.

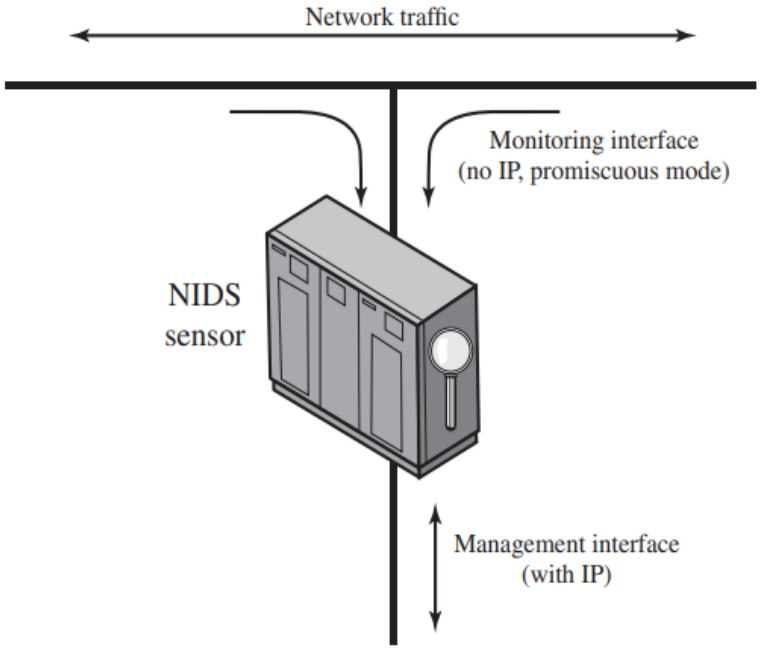

Figure 8.4 **Passive NIDS Sensor**

NIDS Sensor Deployment

Consider an organization with multiple sites, each of which has one or more LANs, with all of the networks interconnected via the Internet or some other WAN technology

Figure 8.5 illustrates a number of possibilities.

In general terms, this configuration is typical of larger organizations. All Internet traffic passes through an external firewall that protects the entire facility.2 Traffic from the outside world, such as customers and vendors that need access to public services, such as Web and mail, is monitored. The external firewall also provides a degree of protection for those parts of the network that should only be accessible by users from other corporate sites. Internal firewalls may also be used to provide more specific protection to certain parts of the network.

NIDS Sensor Deployment

Shashi KS

Instead of placing a NIDS sensor inside the external firewall, the security administrator may choose to place a NIDS sensor between the external firewall and the Internet or WAN (location 2). In this position, the sensor can monitor all network traffic, unfiltered.

A sensor at location 2 has a higher processing burden than any sensor located elsewhere on the site network.

In addition to a sensor at the boundary of the network, on either side of the external firewall, the administrator may configure a firewall and one or more sensors to protect major backbone networks, such as those that support internal servers and database resources (location 3).

Finally, the network facilities at a site may include separate LANs that support user workstations and servers specific to a single department. The administrator could configure a firewall and NIDS sensor to provide additional protection for all of these networks or target the protection to critical subsystems, such as personnel and financial networks (location 4). A sensor used in this latter fashion detects attacks targeting critical systems and resources

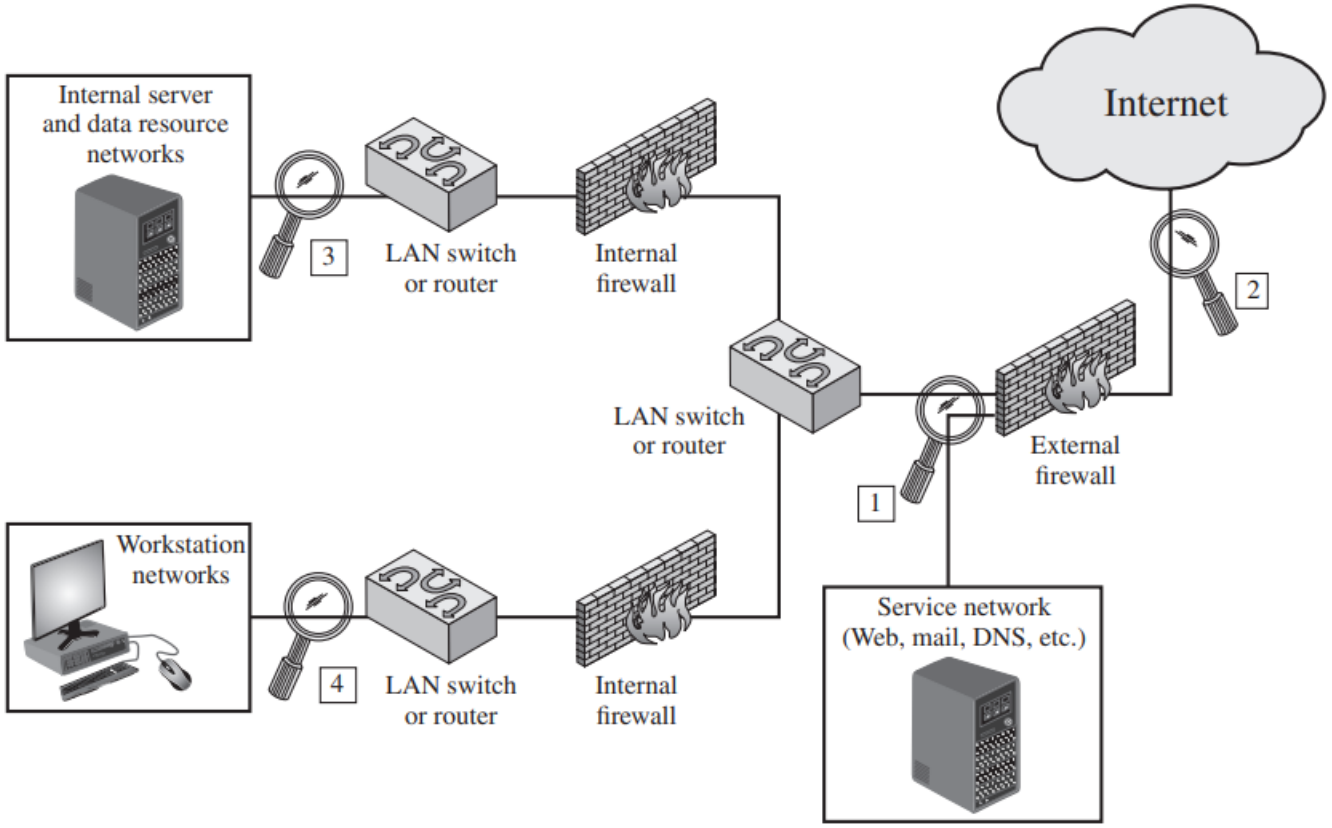

Figure 8.5 **Example of NIDS Sensor Deployment**

DISTRIBUTED OR HYBRID INTRUSION DETECTION

Shashi KS

The concept of communicating IDSs has evolved to schemes that involve distributed systems that cooperate to identify intrusions and to adapt to changing attack profiles. T

The more traditional attack approach is to develop worms and other malicious software that spreads ever more rapidly and to develop other attacks (such as DoS attacks) that strike with overwhelming force before a defense can be mounted. This style of attack is still prevalent. But more recently, attackers have added a quite different approach: Slow the spread of the attack so that it will be more difficult to detect by conventional algorithms

A way to counter such attacks is to develop cooperated systems that can recognize attacks based on more subtle clues and then adapt quickly. In this approach, anomaly detectors at local nodes look for evidence of unusual activity.

For example, a machine that normally makes just a few network connections might suspect that an attack is under way if it is suddenly instructed to make connections at a higher rate. With only this evidence, the local system risks a false positive if it reacts to the suspected attack (say by disconnecting from the network and issuing an alert) but it risks a false negative if it ignores the attack or waits for further evidence. In an adaptive, cooperative system, the local node instead uses a peer-to-peer “gossip” protocol to inform other machines of its suspicion, in the form of a probability that the network is under attack. If a machine receives enough of these messages so that a threshold is exceeded, the machine assumes an attack is under way and responds. The machine may respond locally to defend itself and also send an alert to a central system. An example of this approach is a scheme developed by Intel and referred to as autonomic enterprise security [AGOS06].

Figure 8.6 illustrates the approach.

Shashi KS

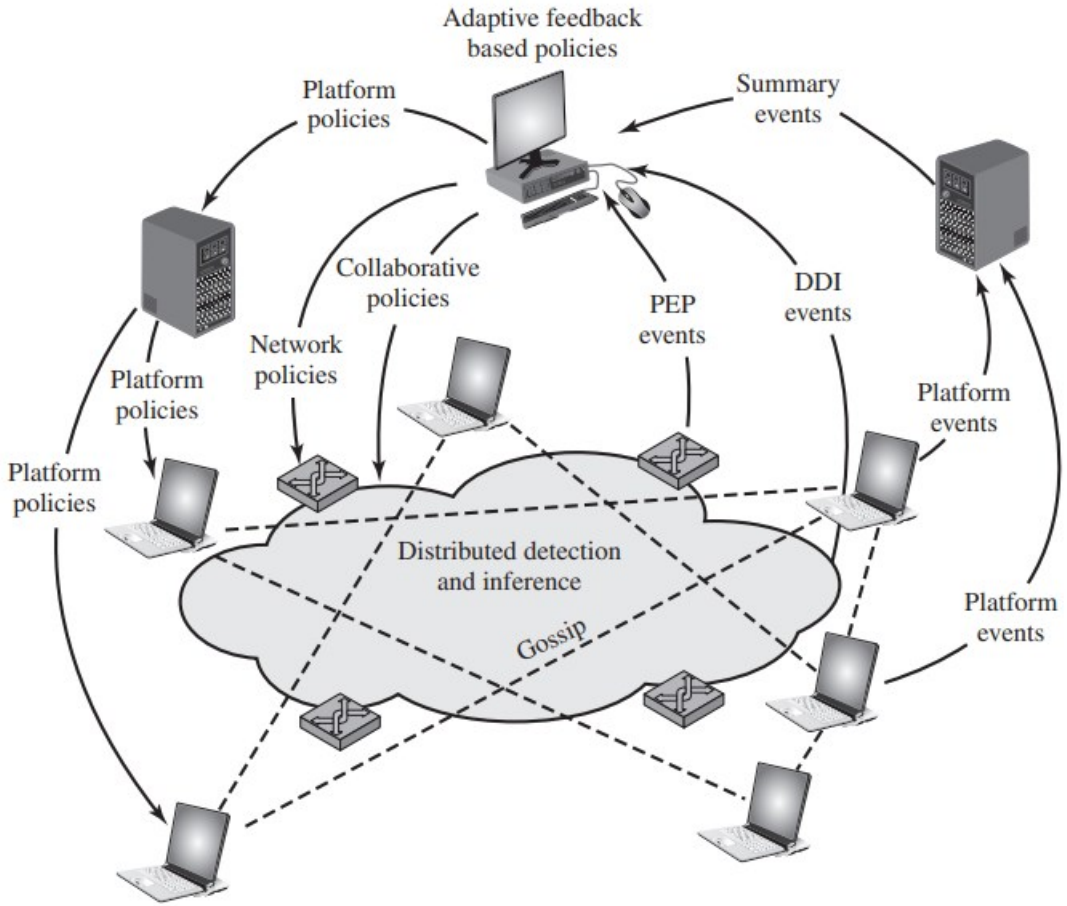

Figure 8.6 Overall Architecture of an Autonomic Enterprise Security System

This approach does not rely solely on perimeter defense mechanisms, such as firewalls, or on individual host-based defenses.

Instead, each end host and each network device (e.g., routers) is considered to be a potential sensor and may have the sensor software module installed.

The sensors in this distributed configuration can exchange information to corroborate the state of the network (i.e., whether an attack is under way).

Principal elements of the distributed intrusion detection system

Shashi KS

- A central system is configured with a default set of security policies.
- Based on input from distributed sensors, these policies are adapted and specific actions are communicated to the various platforms in the distributed system. The device specific policies may include immediate actions to take or parameter settings to be adjusted.
- The central system also communicates collaborative policies to all platforms that adjust the timing and content of collaborative gossip messages.

Three types of input guide the actions of the central system:

• Summary events:
Events from various sources are collected by intermediate collection points such as firewalls, IDSs, or servers that serve a specific segment of the enterprise network. These events are summarized for delivery to the central policy system.

• DDI events:
Distributed detection and inference (DDI) events are alerts that are generated when the gossip traffic enables a platform to conclude that an attack is under way.

• PEP events:
Policy enforcement points (PEPs) reside on trusted, self defending platforms and intelligent IDSs. These systems correlate distributed information, local decisions, and individual device actions to detect intrusions that may not be evident at the host level.

INTRUSION DETECTION EXCHANGE FORMAT

Shashi KS

To facilitate the development of distributed IDSs that can function across a wide range of platforms and environments, standards are needed to support interoperability. Such standards are the focus of the IETF Intrusion Detection Working Group. The purpose of the working group is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response 292 Chapter 8 / Intrusion Detection systems and to management systems that may need to interact with them.

The working group issued the following RFCs in 2007:

Intrusion Detection Message Exchange Requirements (RFC 4766):
This document defines requirements for the Intrusion Detection Message Exchange Format (IDMEF). The document also specifies requirements for a communication protocol for communicating IDMEF.

- The Intrusion Detection Message Exchange Format (RFC 4765):

This document describes a data model to represent information exported by intrusion detection systems and explains the rationale for using this model. An implementation of the data model in the Extensible Markup Language (XML) is presented, an XML Document Type Definition is developed, and examples are provided.

- The Intrusion Detection Exchange Protocol (RFC 4767):

This document describes the Intrusion Detection Exchange Protocol (IDXP), an application-level protocol for exchanging data between intrusion detection entities. IDXP supports mutual-authentication, integrity, and confidentiality over a connection-oriented protocol

Functional components of the model

Figure 8.7 illustrates the key elements of the model on which the intrusion detection message exchange approach is based. This model does not correspond to any particular product or implementation, but its functional components are the key elements of any IDS.

The functional components are as follows:

- **Data source:**

The raw data that an IDS uses to detect unauthorized or undesired activity. Common data sources include network packets, operating system audit logs, application audit logs, and system-generated checksum data.

• Sensor: Collects data from the data source. The sensor forwards events to the analyzer. • Analyzer: The ID component or process that analyzes the data collected by the sensor for signs of unauthorized or undesired activity or for events that might be of interest to the security administrator. In many existing IDSs, the sensor and the analyzer are part of the same component.

• Administrator: The human with overall responsibility for setting the security policy of the organization, and, thus, for decisions about deploying and configuring the IDS. This may or may not be the same person as the operator of the IDS. In some organizations, the administrator is associated with the network or systems administration groups. In other organizations, it is an independent position.

• Manager:

The ID component or process from which the operator manages the various components of the ID system. Management functions typically include sensor configuration, analyzer configuration, event notification management, data consolidation, and reporting.

• Operator: The human that is the primary user of the IDS manager. The operator often monitors the output of the IDS and initiates or recommends further action

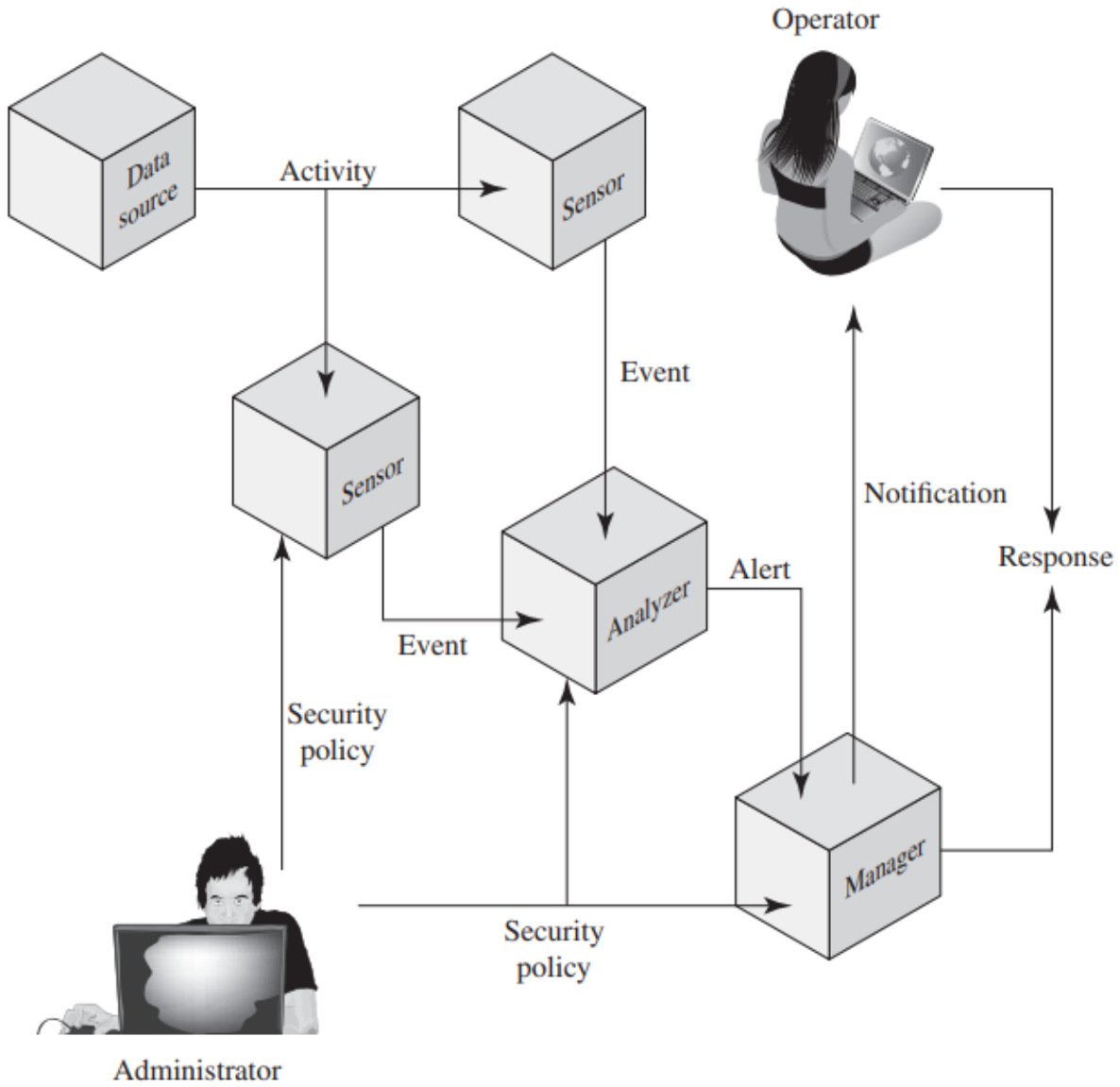

Figure 8.7 Model for Intrusion Detection Message Exchange

Key elements of the intrusion detection model

Shashi KS

In this model, intrusion detection proceeds in the following manner.

- ✓ The sensor monitors data sources looking for suspicious activity, such as network sessions showing unexpected telnet activity, operating system log file entries showing a user attempting to access files to which he or she is not authorized to have access, and application log files showing persistent login failures.

- ✓ The sensor communicates suspicious activity to the analyzer as an event, which characterizes an activity within a given period of time. If the analyzer determines that the event is of interest, it sends an alert to the manager component that contains information about the unusual activity that was detected, as well as the specifics of the occurrence. The manager component issues a notification to the human operator.

- ✓ A response can be initiated automatically by the manager component or by the human operator. Examples of responses include logging the activity; recording the raw data (from the data source) that characterized the event; terminating a network, user, or application session; or altering network or system access controls.

- ✓ The security policy is the predefined, formally documented statement that defines what activities are allowed to take place on an organization's network or on particular hosts to support the organization's requirements.

- ✓ This includes, but is not limited to, which hosts are to be denied external network access.
- ✓ The specification defines formats for event and alert messages, message types, and exchange protocols for communication of intrusion detection information.