CMPE 209 Network Security

Chapter 8. Intrusion Detection

Dr. Younghee Park

SAN JOSÉ STATE
UNIVERSITY

Examples of Intrusion

- Remote root compromise
- Web server defacement
- Guessing/cracking passwords
- Copying databases containing credit card numbers
- Viewing sensitive data without authorization
- Running a packet sniffer
- Distributing pirated software
- Using an unsecured modem to access internal network
- Impersonating an executive to get information
- Using an unattended workstation

Intruder Behavior

Target acquisition and information gathering	Initial access	Privilege escalation
Information gathering or system exploit	Maintaining access	Covering tracks

(a) Target Acquisition and Information Gathering

- Explore corporate website for information on corporate structure, personnel, key systems, as well as details of specific web server and OS used.
- Gather information on target network using DNS lookup tools such as dig, host, and others; and query WHOIS database.
- Map network for accessible services using tools such as NMAP.
- Send query email to customer service contact, review response for information on mail client, server, and OS used, and also details of person responding.
- Identify potentially vulnerable services, eg vulnerable web CMS.

(b) Initial Access

- Brute force (guess) a user's web content management system (CMS) password.
- Exploit vulnerability in web CMS plugin to gain system access.
- Send spear-phishing email with link to web browser exploit to key people.

(c) Privilege Escalation

- Scan system for applications with local exploit.
- Exploit any vulnerable application to gain elevated privileges.
- Install sniffers to capture administrator passwords.
- Use captured administrator password to access privileged information.

(d) Information Gathering or System Exploit

- Scan files for desired information.
- Transfer large numbers of documents to external repository.
- Use guessed or captured passwords to access other servers on network.

(e) Maintaining Access

- Install remote administration tool or rootkit with backdoor for later access.
- Use administrator password to later access network.
- Modify or disable anti-virus or IDS programs running on system.

(f) Covering Tracks

- Use rootkit to hide files installed on system.
- Edit logfiles to remove entries generated during the intrusion.

Table 8.1

Examples of Intruder Behavior

(Table can be found on pages 271-272 in textbook.)

Definitions from RFC 2828 (Internet Security Glossary)

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)

Host-based IDS (HIDS)

Monitors the characteristics of a single host for suspicious activity

Network-based IDS (NIDS)

Monitors network traffic 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

Comprises three logical components:

- **Sensors - collect data**
- **Analyzers - determine if intrusion has occurred**
- **User interface - view output or control system behavior**

IDS Requirements

Run continually	Be fault tolerant	Resist subversion
Impose a minimal overhead on system	Configured according to system security policies	Adapt to changes in systems and users
Scale to monitor large numbers of systems	Provide graceful degradation of service	Allow dynamic reconfiguration

Analysis Approaches

Anomaly detection

- Involves the collection of data relating to the behavior of legitimate users over a period of time
- Current observed behavior is analyzed to determine whether this behavior is that of a legitimate user or that of an intruder

Signature/Heuristic detection

- Uses a set of known malicious data patterns or attack rules that are compared with current behavior
- Also known as misuse detection
- Can only identify known attacks for which it has patterns or rules

Anomaly Detection

A variety of classification approaches are used:

Statistical	Knowledge based	Machine-learning
• Analysis of the observed behavior using univariate, multivariate, or time-series models of observed metrics	• Approaches use an expert system that classifies observed behavior according to a set of rules that model legitimate behavior	• Approaches automatically determine a suitable classification model from the training data using data mining techniques

Signature or Heuristic Detection

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

Widely used in anti-virus products, network traffic scanning proxies, and in NIDS

Rule-based heuristic identification

Involves the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses

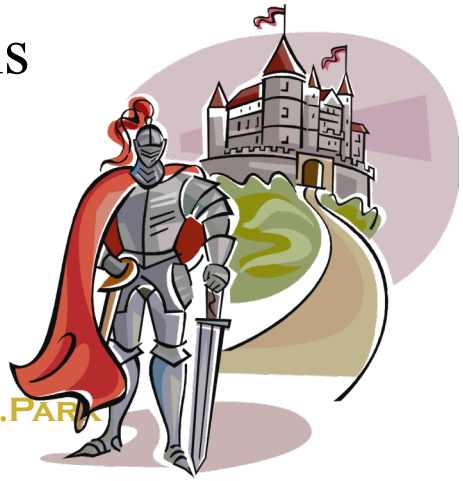
Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage

Typically rules used are specific

SNORT is an example of a rule-based NIDS

Host-Based Intrusion Detection (HIDS)

- Adds a specialized layer of security software to vulnerable or sensitive systems
- Can use either anomaly or signature and heuristic approaches
- Monitors activity to detect suspicious behavior
  - Primary purpose is to detect intrusions, log suspicious events, and send alerts
  - Can detect both external and internal intrusions



Data Sources and Sensors

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

Common data sources include:

- System call traces
- Audit (log file) records
- File integrity checksums
- Registry access

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

comctl32
kernel32
msvcpp
msvcrt
mswsock
ntdll
ntoskrnl
user32
ws2_32

Table 8.2

Linux System Calls and Windows DLLs Monitored

(Table can be found on page 280 in the textbook)

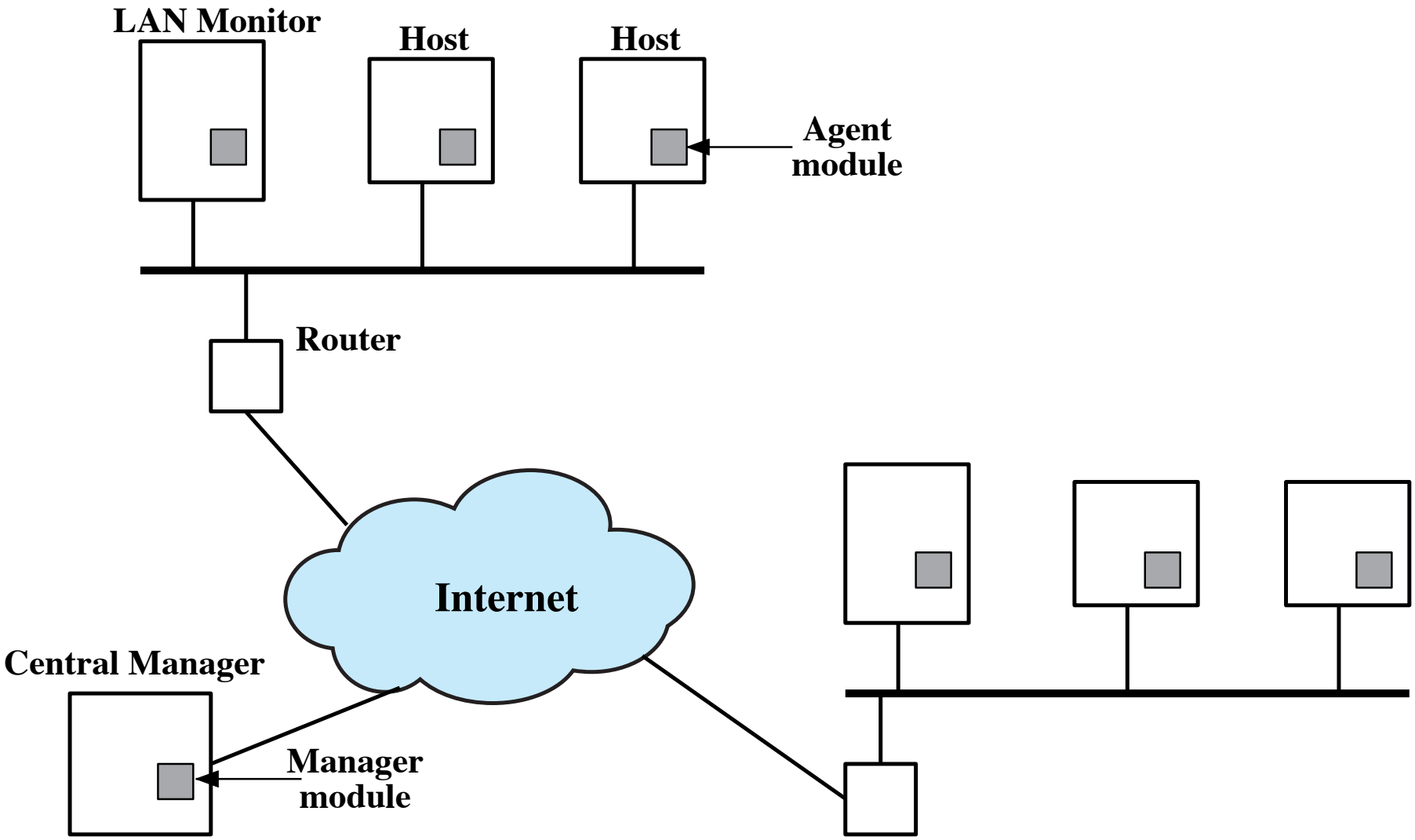
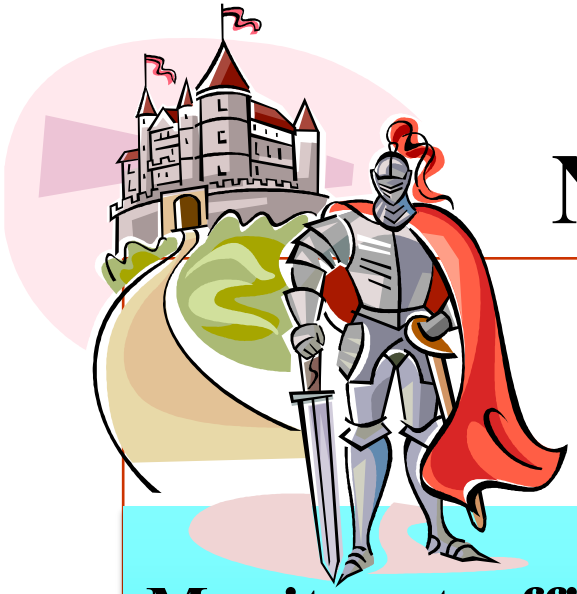


Figure 8.2 Architecture for Distributed Intrusion Detection



Network-Based IDS (NIDS)

Monitors traffic at selected points on a network

Examines traffic packet by packet in real or close to real time

May examine network, transport, and/or application-level protocol activity

Comprised of a number of sensors, one or more servers for NIDS management functions, and one or more management consoles for the human interface

Analysis of traffic patterns may be done at the sensor, the management server or a combination of the two

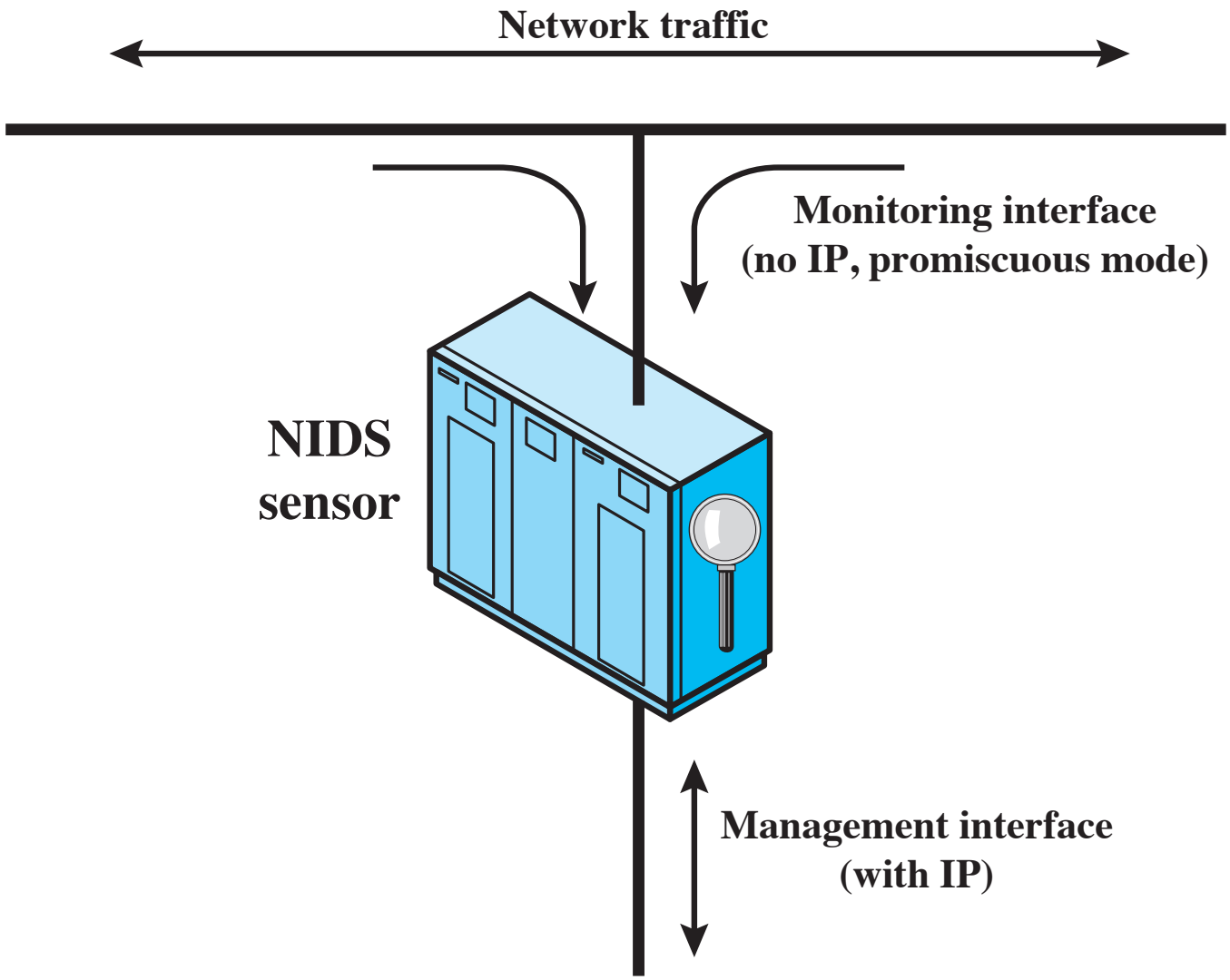


Figure 8.4 Passive NIDS Sensor

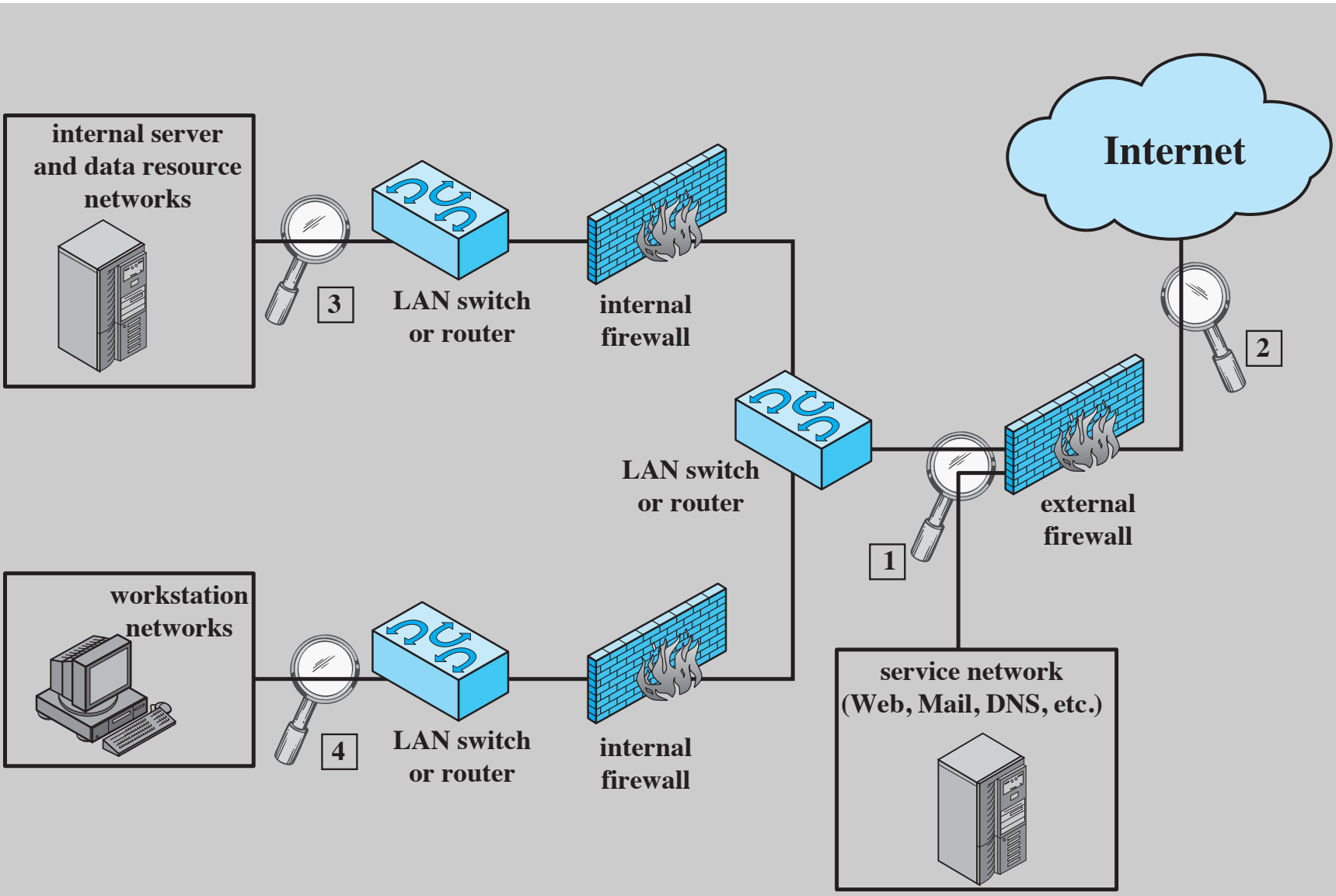


Figure 8.5 Example of NIDS Sensor Deployment

Intrusion Detection Techniques

Attacks suitable for Signature detection

- Application layer reconnaissance and attacks
- Transport layer reconnaissance and attacks
- Network layer reconnaissance and attacks
- Unexpected application services
- Policy violations

Attacks suitable for Anomaly detection

- Denial-of-service (DoS) attacks
- Scanning
- Worms

Logging of Alerts

- Typical information logged by a NIDS sensor includes:
  - Timestamp
  - Connection or session ID
  - Event or alert type
  - Rating
  - Network, transport, and application layer protocols
  - Source and destination IP addresses
  - Source and destination TCP or UDP ports, or ICMP types and codes
  - Number of bytes transmitted over the connection
  - Decoded payload data, such as application requests and responses
  - State-related information

Honeypots

- **Decoy systems designed to:**
  - Lure a potential attacker away from critical systems
  - Collect information about the attacker's activity
  - Encourage the attacker to stay on the system long enough for administrators to respond
- **Systems are filled with fabricated information that a legitimate user of the system wouldn't access**
- **Resources that have no production value**
  - Therefore incoming communication is most likely a probe, scan, or attack
  - Initiated outbound communication suggests that the system has probably been compromised

Snort

Snort is an open source, highly configurable and portable host-based or network-based IDS. Snort is referred to as a lightweight IDS, which has the following characteristics:

- Easily deployed on most nodes (host, server, router) of a network
- Efficient operation that uses small amount of memory and processor time
- Easily configured by system administrators who need to implement a specific security solution in a short amount of time

Snort

Snort can perform real-time packet capture, protocol analysis, and content searching and matching. Snort can detect a variety of attacks and probes, based on a set of rules configured by a system administrator.

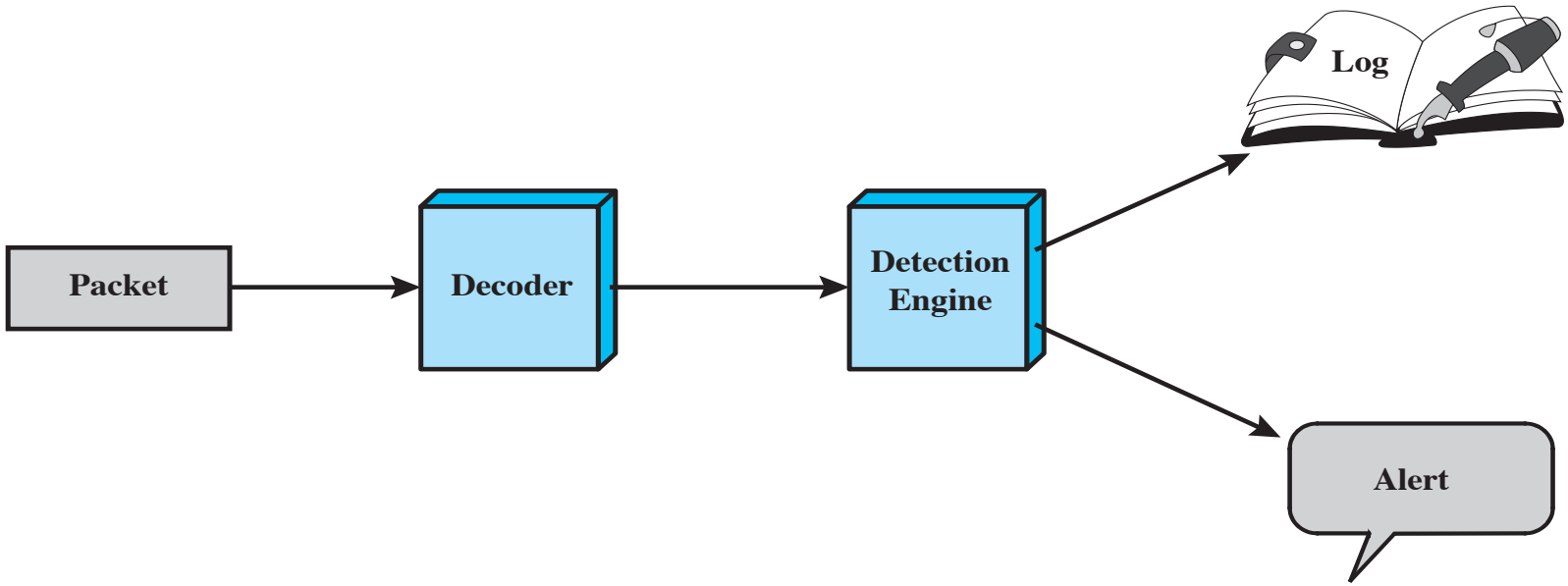


Figure 8.9 Snort Architecture

Snort uses a simple, flexible rule definition language that generates the rules used by the detection engine. Although the rules are simple and straightforward to write, they are powerful enough to detect a wide variety of hostile or suspicious traffic. Each rule consists of a fixed header and zero or more options (Figure 8.10).

Action	Protocol	Source IP address	Source Port	Direction	Dest IP address	Dest Port

(a) Rule Header

Option Keyword	Option Arguments	• • •

(b) Options

Figure 8.10 Snort Rule Formats

Snort Rules

The header has the following elements:

- **Action:** The rule action tells Snort what to do when it finds a packet that matches the rule criteria. Table 8.3 lists the available actions. The last three actions in the list (drop, reject, sdrop) are only available in inline mode.
- **Protocol:** Snort proceeds in the analysis if the packet protocol matches this field. The current version of Snort (2.9) recognizes four protocols: TCP, UDP, ICMP, and IP. Future releases of Snort will support a greater range of protocols.

Snort Rules

- **Source IP address:** Designates the source of the packet. The rule may specify a specific IP address, any IP address, a list of specific IP addresses, or the negation of a specific IP address or list. The negation indicates that any IP address other than those listed is a match.
- **Source port:** This field designates the source port for the specified protocol (e.g., a TCP port). Port numbers may be specified in a number of ways, including specific port number, any ports, static port definitions, ranges, and by negation.

Snort Rules

- **Direction:** This field takes on one of two values: unidirectional (->) or bidirectional (<->). The bidirectional option tells Snort to consider the address/port pairs in the rule as either source followed by destination or destination followed by source. The bidirectional option enables Snort to monitor both sides of a conversation.
- **Destination IP address:** Designates the destination of the packet.

Snort Rules

- **Destination port:** Designates the destination port.
- Following the rule header may be one or more rule options. Each option consists of an option keyword, which defines the option; followed by arguments, which specify the details of the option. In the written form, the set of rule options is separated from the header by being enclosed in parentheses. Snort rule options are separated from each other using the semicolon (;) character. Rule option keywords are separated from their arguments with a colon (:) character.

Table 8.3

Snort Rule Actions

Action	Description
alert	Generate an alert using the selected alert method, and then log the packet.
log	Log the packet.
pass	Ignore the packet.
activate	Alert and then turn on another dynamic rule.
dynamic	Remain idle until activated by an activate rule , then act as a log rule.
drop	Make iptables drop the packet and log the packet.
reject	Make iptables drop the packet, log it, and then send a TCP reset if the protocol is TCP or an ICMP port unreachable message if the protocol is UDP.
sdrop	Make iptables drop the packet but does not log it.

meta-data
msg Defines the message to be sent when a packet generates an event. **reference** Defines a link to an external attack identification system, which provides additional information. **classtype** Indicates what type of attack the packet attempted.
payload
content Enables Snort to perform a case-sensitive search for specific content (text and/or binary) in the packet payload. **depth** Specifies how far into a packet Snort should search for the specified pattern. Depth modifies the previous content keyword in the rule. **offset** Specifies where to start searching for a pattern within a packet. Offset modifies the previous content keyword in the rule. **nocase** Snort should look for the specific pattern, ignoring case. Nocase modifies the previous content keyword in the rule.
non-payload
ttl Check the IP time-to-live value. This option was intended for use in the detection of traceroute attempts. **id** Check the IP ID field for a specific value. Some tools (exploits, scanners and other odd programs) set this field specifically for various purposes, for example, the value 31337 is very popular with some hackers. **dsize** Test the packet payload size. This may be used to check for abnormally sized packets. In many cases, it is useful for detecting buffer overflows. **flags** Test the TCP flags for specified settings. **seq** Look for a specific TCP header sequence number. **icmp-id** Check for a specific ICMP ID value. This is useful because some covert channel programs use static ICMP fields when they communicate. This option was developed to detect the stacheldraht DDoS agent.
post-detection
logto Log packets matching the rule to the specified filename. **session** Extract user data from TCP Sessions. There are many cases where seeing what users are typing in telnet, rlogin, ftp, or even web sessions is very useful.

Table 8.4

Examples of Snort Rule Options

(Table can be found on page 299 in textbook.)

Snort Rule Example

(Example 1)

log udp any any->192.168.1.0/24 1:1024

This means that it logs udp traffic coming from any port and destination ports ranging from 1 to 1024

Snort Rule Example 2

```
activate tcp !$HOME_NET any -> $HOME_NET 143 (flags:PA; \
    content:"|E8C0FFFFFF|/bin"; activates:1;  \
    msg:"IMAP buffer overflow!";)
dynamic tcp !$HOME_NET any -> $HOME_NET 143 (activated_by:1; count:50;)
```

These rules tell Snort to alert when it detects an IMAP buffer overflow and collect the next 50 packets headed for port 143 coming from outside $HOME_NET headed to $HOME_NET. If the buffer overflow happened and was successful, there's a very good possibility that useful data will be contained within the next 50 (or whatever) packets going to that same service port on the network, so there's value in collecting those packets for later analysis.

Snort Rules

- Snort has many configuration variables and options, but the two most important ones are $HOME_NET and $EXTERNAL_NET. $HOME_NET is a variable that defines the network or networks you are trying to protect, while $EXTERNAL_NET is the external, untrusted networks to which you are connected. These variables are used in virtually all rules to specify criteria for the source and destination of a packet.

Summary

- Intrusion detection
- Analysis approaches
  - Anomaly detection
  - Signature or heuristic detection
- Distributed or hybrid intrusion detection
- Honeypots
- Host-based intrusion detection
  - Data sources and sensors
  - Anomaly HIDS
  - Signature or heuristic HIDS
  - Distributed HIDS
- Network-based intrusion detection
  - Types of network sensors
  - NIDS sensor deployment
  - Intrusion detection techniques
  - Logging of alerts
- Example system: Snort
  - Snort architecture
  - Snort rules