Chapter 9 (9.1, 9.2, 9.3, 9.6)

The Need for Firewalls:

Consider a network with hundreds or even thousands of systems, running various operating systems, such as different versions of Windows, MacOS, and Linux. When a security flaw is discovered, each potentially affected system must be upgraded to fix that flaw. This requires scalable configuration management and aggressive patching to function effectively. While difficult, this is possible and is necessary if only host-based security is used. A better alternative is firewall.

The firewall is inserted between the premises network and the Internet to establish a controlled link and to erect an outer security wall or perimeter. The aim of this perimeter is to protect the premises network from Internet-based attacks and to provide a single choke point where security and auditing can be imposed. The firewall may be a single computer system or a set of two or more systems that cooperate to perform the firewall function. The firewall, then, provides an additional layer of defense, insulating the internal systems from external networks.

Firewall Design Goals:

- All traffic from inside to outside, and vice versa, must pass through the firewall
- Only authorized traffic as defined by the local security policy will be allowed to pass
- The firewall itself is immune to penetration

Firewall Access Policy:

It is a critical component in the planning and implementation of a firewall. This lists the types of traffic authorized to pass through the firewall, such as: following are the firewall filters:

- **IP Address and Protocol Values**: Controls access based on the source or destination addresses and port numbers, inbound or outbound traffic, and others. This type of filtering is used by packet filter and stateful inspection firewalls. It is typically used to limit access to specific services.
- **Application Protocol:** Controls access on the basis of authorized application protocol data, for example, checking SMTP e-mail for spam, or HTPP Web requests to authorized sites only.
- **User Identity:** Controls access based on the user's identity, typically for inside users who identify themselves using some form of secure authentication technology.
- **Network Activity**: Controls access based on considerations such as the time or request, for example, only in business hours; rate of requests, for example, to detect scanning attempts; or other activity patterns.

Firewall Capabilities:

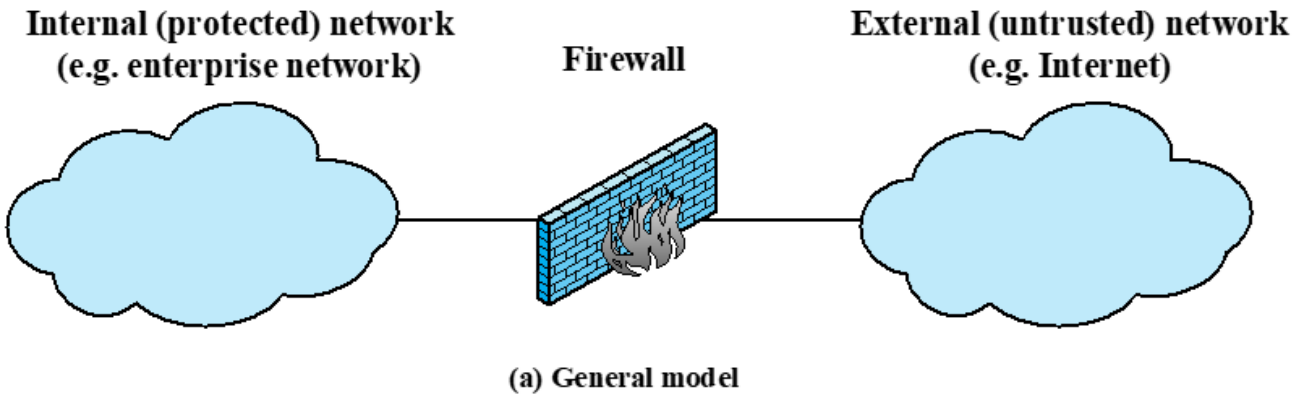
1. A **firewall defines a single choke point** that attempts to keep unauthorized users out of the protected network, prohibit potentially vulnerable services from entering or leaving the network, and provide protection from various kinds of **IP spoofing and routing attacks**.
2. A firewall **provides a location for monitoring security-related events**. Audits and alarms can be implemented on the firewall system.

3. A firewall is a convenient platform for **several Internet functions that are not security related**. These include a **network address translator**, which maps local addresses to Internet addresses, and a network management function that audits or logs Internet usage.
4. A firewall can **serve as the platform for IPSec.** Using the tunnel mode capability, the firewall can be used to implement **virtual private networks.**

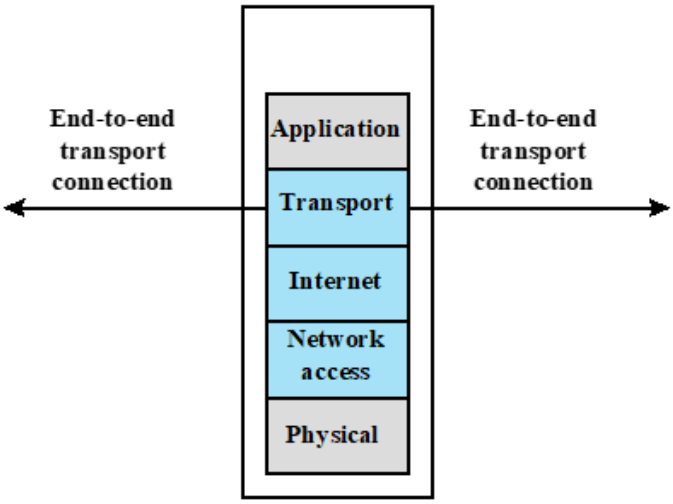
Firewall Limitations:

1. The firewall cannot protect against attacks that bypass the firewall.
2. The firewall may not protect fully against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.
3. An improperly secured wireless LAN may be accessed from outside the organization.
4. A laptop, PDA, or portable storage device may be used and infected outside the corporate network and then attached and used internally.

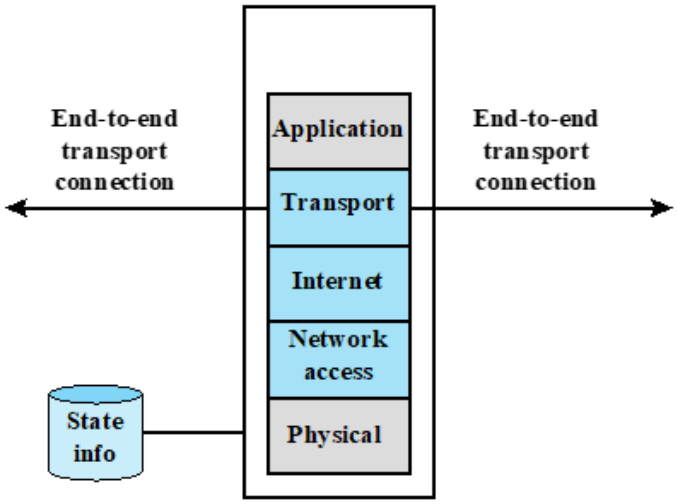
Types of Firewalls:



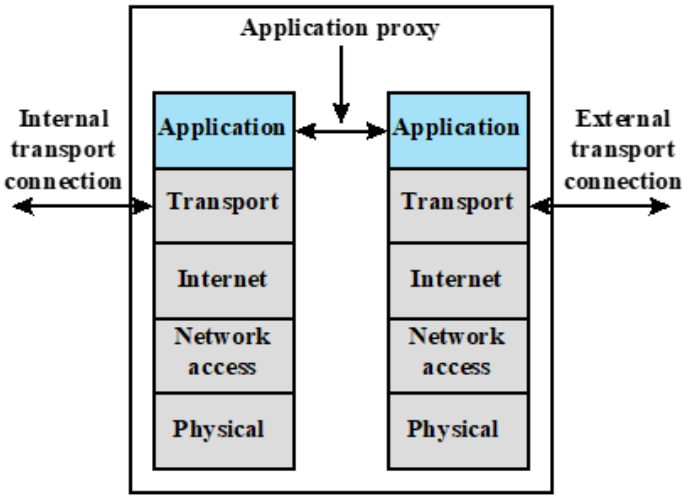
(a) General model



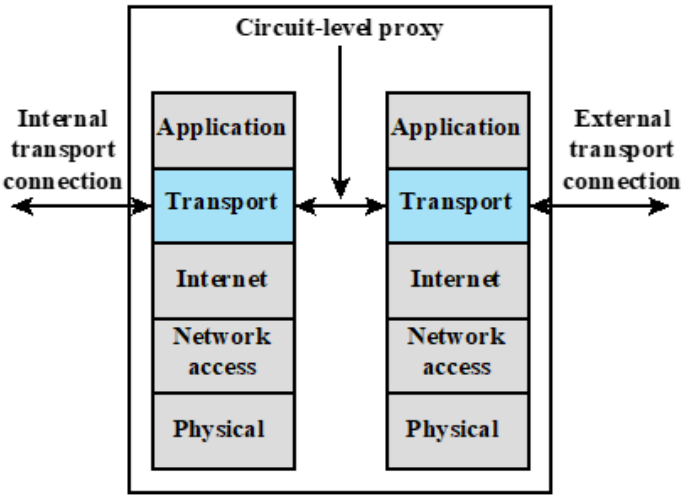
(b) Packet filtering firewall



(c) Stateful inspection firewall



(d) Application proxy firewall



(e) Circuit-level proxy firewall

A **packet filtering firewall** applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet (Figure 9.1b). The firewall is typically configured to filter packets going in both directions (from and to the internal network). Filtering rules are based on information contained in a network packet:

- Source IP address
- Destination IP address
- Source and destination transport-level address
- IP protocol field
- Interface

Table 9.1 Packet-Filtering Examples

Rule	Direction	Src address	Dest addresss	Protocol	Dest port	Action
1	In	External	Internal	TCP	25	Permit
2	Out	Internal	External	TCP	>1023	Permit
3	Out	Internal	External	TCP	25	Permit
4	In	External	Internal	TCP	>1023	Permit
5	Either	Any	Any	Any	Any	Deny

The intent of each rule is:

1. Inbound mail from an external source is allowed (port 25 is for SMTP incoming).
2. This rule is intended to allow a response to an inbound SMTP connection.
3. Outbound mail to an external source is allowed.
4. This rule is intended to allow a response to an outbound SMTP connection.
5. This is an explicit statement of the default policy. All rule sets include this rule implicitly as the last rule.

There are several problems with this rule set.

Rule 4 allows external traffic to any destination port above 1023. As an example of an exploit of this rule, an external attacker can open a connection from the attacker's port 5150 to an internal Web proxy server on port 8080. This is supposed to be forbidden and could allow an attack on the server. To counter this attack, the firewall rule set can be

configured with a source port field for each row. For rules 2 and 4, the source port is set to 25; for rules 1 and 3, the source port is set to 71023

But a vulnerability remains. Rules 3 and 4 are intended to specify that any inside host can send mail to the outside. The problem with this rule is that the use of port 25 for SMTP receipt is only a default; an outside machine could be configured to have some other application linked to port 25. As the revised rule 4 is written, an attacker could gain access to internal machines by sending packets with a TCP source port number of 25. To counter this threat, we can add an ACK flag field to each row. For rule 4, the field would indicate that the ACK flag must be set on the incoming packet. Rule 4 would now look like this:

Rule	Direction	Src address	Src port	Dest address	Protocol	Dest port	Flag	Action
4	In	External	25	Internal	TCP	>1023	ACK	Permit

Packet-Filtering Weaknesses:

1. **Limited effectiveness against certain types of attacks**: Packet filtering firewalls are only effective against network-based attacks that operate at the network or transport layer of the OSI model. They cannot protect against application-layer attacks, such as SQL injection or cross-site scripting, that target vulnerabilities in specific applications or protocols.
2. **Difficulty creating and maintaining rules**: Configuring packet filtering rules can be a complex and time-consuming process, especially in large or dynamic networks. Maintaining these rules can also be challenging, as the network environment may change over time and new rules may need to be added or existing rules may need to be updated to reflect these changes.
3. **Potential for false positives and false negatives**: Packet filtering firewalls are based on a set of predefined rules that determine which packets are allowed to pass through the firewall and which are blocked. If these rules are not configured correctly, it is possible for the firewall to block legitimate traffic or allow malicious traffic to pass through.
4. **Performance degradation**: Because packet filtering firewalls must inspect each packet that passes through the network, they can potentially impact the performance of the network if the volume of traffic is high.
5. **Inability to inspect encrypted traffic**: Packet filtering firewalls can only inspect the headers of packets, not the actual payloads. This means that they are unable to inspect encrypted traffic, such as SSL- or TLS-encrypted traffic, to determine if it is malicious.

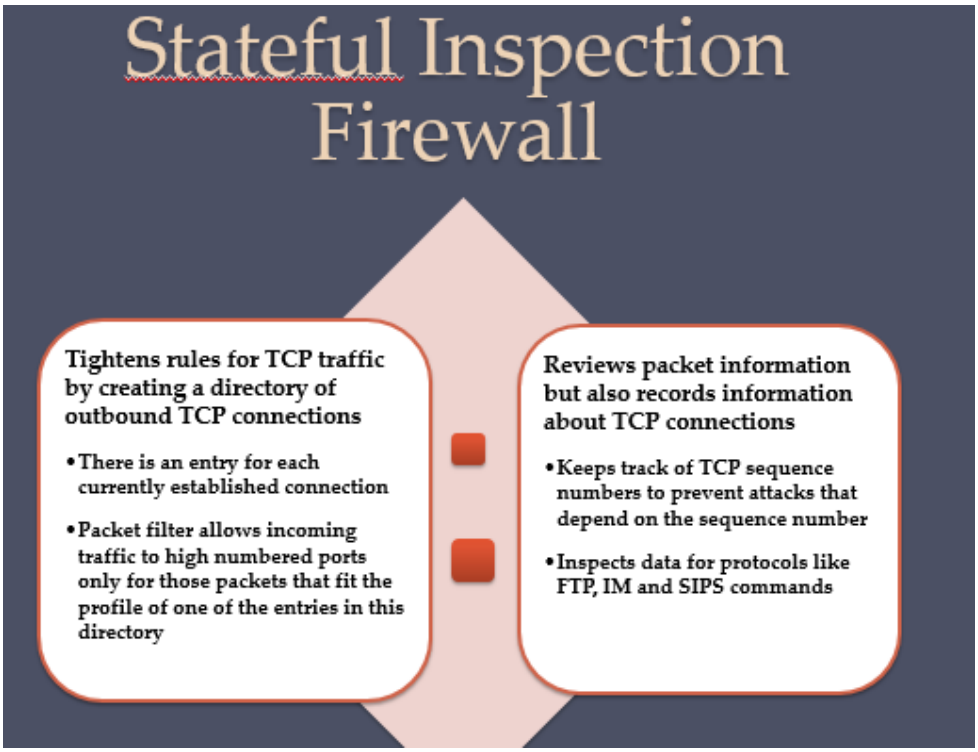
Attacks on Packet-filtering and its Countermeasures:

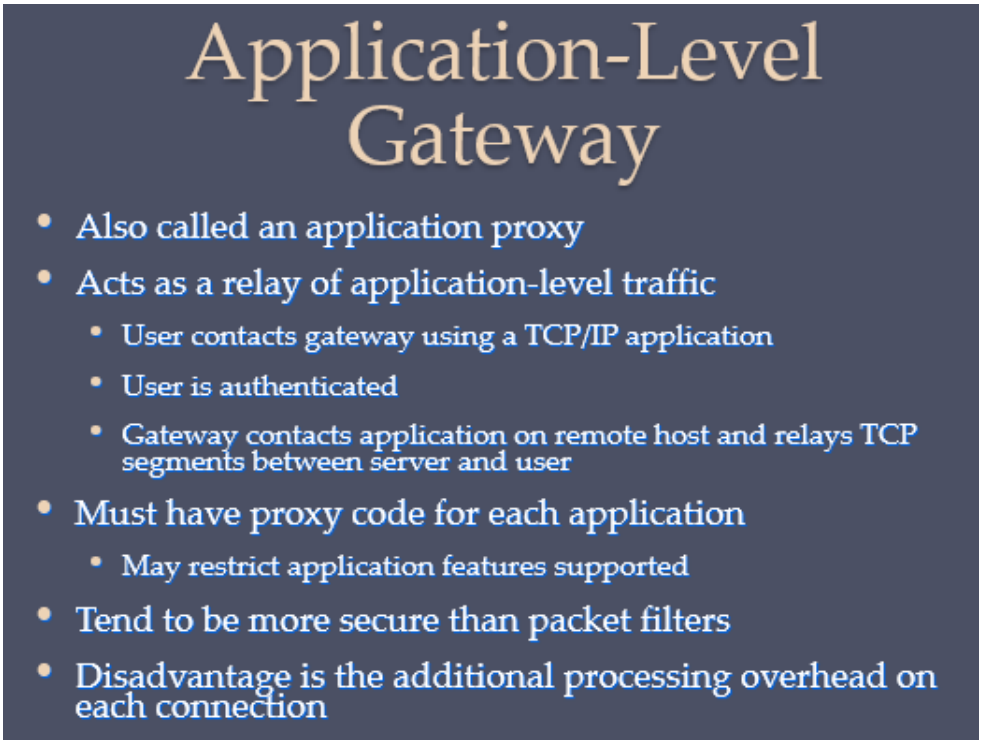
- **IP address spoofing:** The intruder transmits packets from the outside with a source IP address field containing an address of an internal host. The countermeasure is to discard packets with an inside source address if the packet arrives on an external interface. In fact, this countermeasure is often implemented at the router external to the firewall.
- **Source routing attacks**: The source station specifies the route that a packet should take as it crosses the Internet, in the hopes that this will bypass security measures that do not analyze the source routing information. A countermeasure is to discard all packets that use this option.
- **Tiny fragment attacks**: The intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into a separate packet fragment. Typically, a packet filter will make a filtering decision on the first fragment of a packet. All subsequent fragments of that packet are filtered out solely on the

basis that they are part of the packet whose first fragment was rejected. A tiny fragment attack can be defeated by enforcing a rule that the first fragment of a packet must contain a predefined minimum amount of the transport header. If the first fragment is rejected, the filter can remember the packet and discard all subsequent fragments

Difference between Stateful and Stateless Firewall:

A **stateful firewall** tracks the state of network connections and allows traffic based on the context of the TCP connections, while a **stateless firewall** allows or blocks traffic based solely on the information in individual packets. In other words, a **stateful firewall** is more sophisticated and can make more informed decisions about network traffic, while a **stateless firewall** is simpler and less effective at blocking unwanted traffic.





Application-level gateways tend to be more secure than packet filters. Rather than trying to deal with the numerous possible combinations that are to be allowed and forbidden at the TCP and IP level, the application-level gateway need only scrutinize a few allowable applications. In addition, it is easy to log and audit all incoming traffic at the application level.

A prime disadvantage of this type of gateway is the additional processing overhead on each connection. In effect, there are two spliced connections between

the end users, with the gateway at the splice point, and the gateway must examine and forward all traffic in both directions**.**

Circuit-level gateway

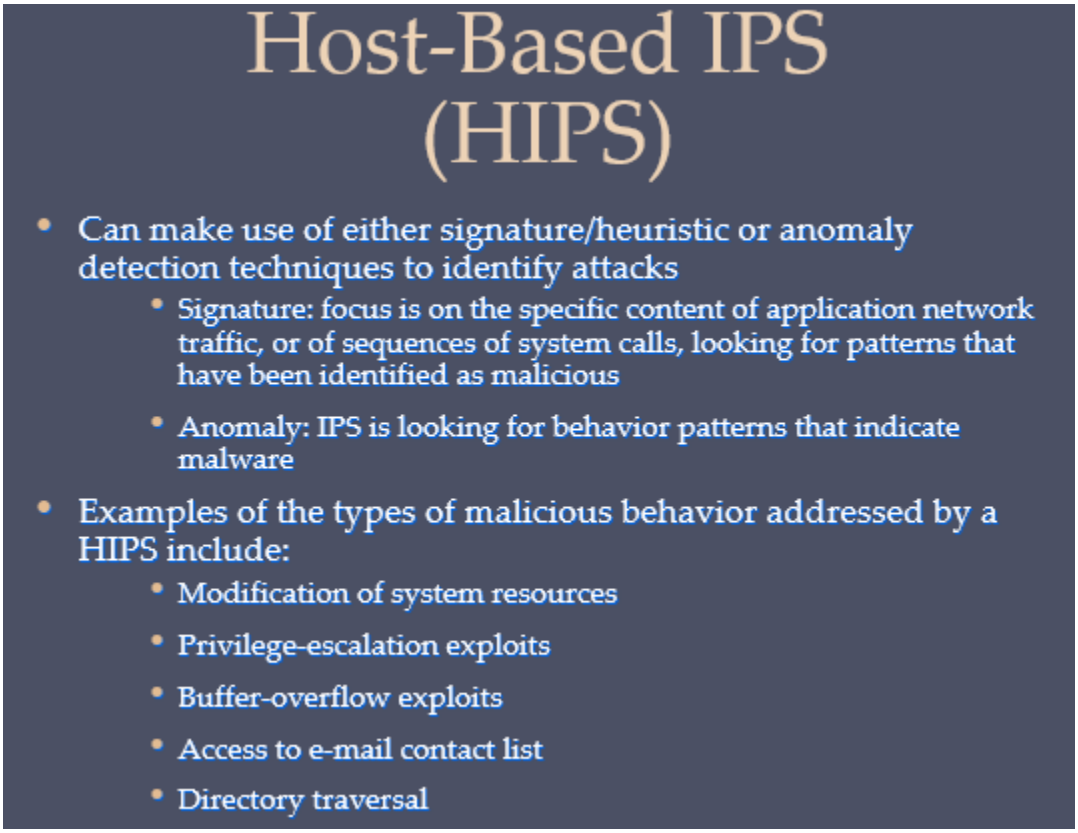
As with an application gateway, a **circuit-level gateway** does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

A typical use of circuit-level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application-level or proxy service on inbound connections and circuit-level functions for outbound connections. In this configuration, the gateway can incur the processing overhead of examining incoming application data for forbidden functions but does not incur that overhead on outgoing data.

Intrusion Prevention Systems (IPS)

It is an extension of an IDS that includes the capability to attempt to block or prevent detected malicious activity. Once an IDS has detected malicious activity, it can respond by modifying or blocking network packets across a perimeter or into a host, or by modifying or blocking system calls by programs running on a host. Thus, a network IPS can block traffic, as a firewall does, but makes use of the types of algorithms developed for IDSs to determine when to do so.

1. **Host-Based IPS (HIPS):**



- **Modification of system resources**: Rootkits, Trojan horses, and backdoors operate by changing system resources, such as libraries, directories, registry settings, and user accounts.

- **Privilege-escalation exploits**: These attacks attempt to give ordinary users root access.
- **Buffer-overflow exploits**: Attackers exploit buffer overflow issues by overwriting the memory of an application
- **Access to e-mail contact list**: Many worms spread by mailing a copy of themselves to addresses in the local system's e-mail address book.
- **Directory traversal**: A directory traversal vulnerability in a Web server allows the hacker to access files outside

Areas for which a HIPS typically offers desktop protection:

- **System calls:** The kernel controls access to system resources such as memory, I/O devices, and processor. To use these resources, user applications invoke system calls to the kernel. Any exploit code will execute at least one system call. The HIPS can be configured to examine each system call for malicious characteristics.
- **File system access**: The HIPS can ensure that file access system calls are not malicious and meet established policy.
- **System registry settings**: The registry maintains persistent configuration information about programs and is often maliciously modified to extend the life of an exploit. The HIPS can ensure that the system registry maintains its integrity.
- **Host input/output**: I/O communications, whether local or network-based, can propagate exploit code and malware. The HIPS can examine and enforce proper client interaction with the network and its interaction with other devices

2. Network-Based IPS (NIPS):

A network-based IPS (NIPS) is inline NIDS with the authority to discard packets and tear down TCP connections. As with a NIDS, a NIPS makes use of techniques such as signature detection and anomaly detection. General methods used by a NIPS device to identify malicious packets are:

a. **Pattern matching:** Scans incoming packets for specific byte sequences (the signature) stored in a database of known attacks
b. **Stateful matching:** Scans for attack signatures in the context of a traffic stream rather than individual packets
c. **Protocol anomaly:** Looks for deviation from standards set forth in RFCs
d. **Traffic anomaly:** Watches for unusual traffic activities, such as a flood of UDP packets or a new service appearing on the network
e. **Statistical anomaly:** Develops baselines of normal traffic activity and throughput, and alerts on deviations from those baselines

3. Distributed or Hybrid IPS:

The final category of IPS is in a distributed or hybrid approach. This gathers data from a large number of host and network-based sensors, relays this intelligence to a central analysis system able to correlate, and analyze the data, which can then return updated signatures and behavior patterns to enable all of the coordinated systems to respond and defend against malicious behavior. A number of such systems have been proposed. One of the best known is the digital immune system.

Digital Immune System

- Comprehensive defense against malicious behavior caused by malware
- Developed by IBM and refined by Symantec
- Motivation for this development includes the rising threat of Internet-based malware, the increasing speed of its propagation provided by the Internet, and the need to acquire a global view of the situation
- Success depends on the ability of the malware analysis system to detect new and innovative malware strains

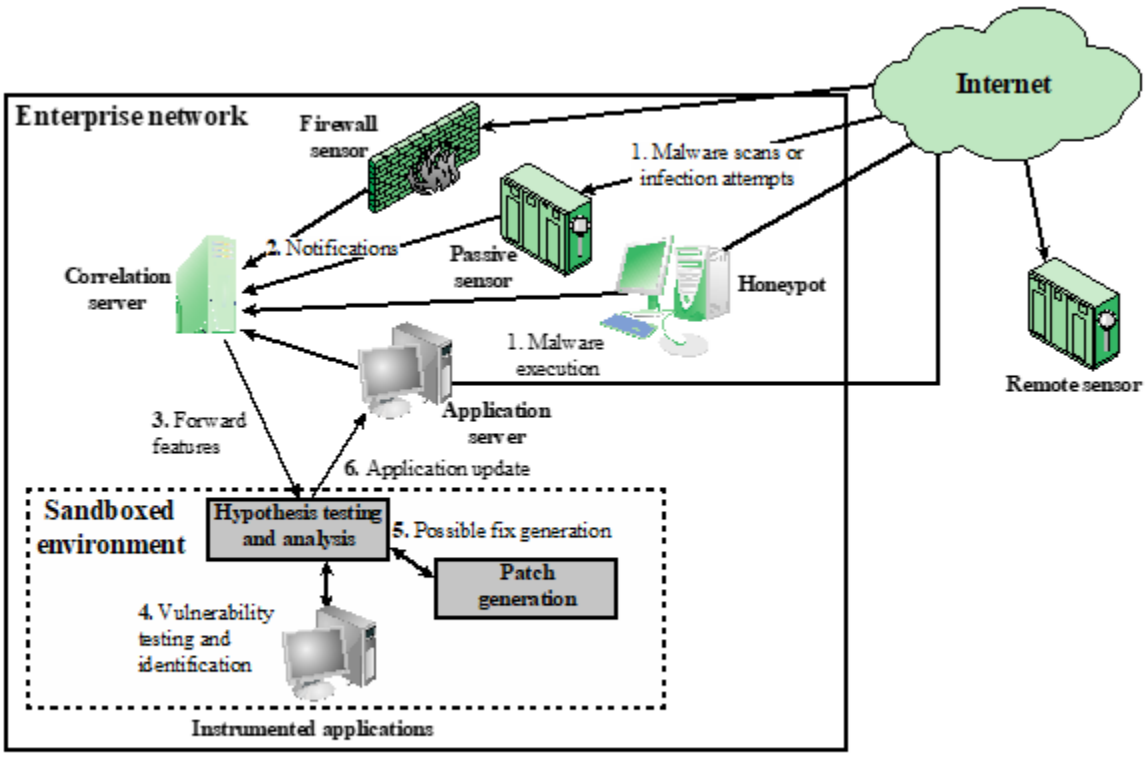


Figure 9.5 Placement of Malware Monitors (adapted from [SIDI05])

1. Sensors deployed at various network and host locations detect potential malware scanning, infection or execution. The sensor logic can also be incorporated in IDS sensors.
2. The sensors send alerts and copies of detected malware to a central server, which correlates and analyzes this information. The correlation server determines the likelihood that malware is being observed and its key characteristics.

3. The server forwards its information to a protected environment, where the potential malware may be sandboxed for analysis and testing.
4. The protected system tests the suspicious software against an appropriately instrumented version of the targeted application to identify the vulnerability.
5. The protected system generates one or more software patches and tests these.
6. If the patch is not susceptible to the infection and does not compromise the application's functionality, the system sends the patch to the application host to update the targeted application.