Security at Network Layer

Introduction

Network layer security controls have been used frequently for securing communications, particularly over shared networks such as the Internet because they can provide protection for many applications at once without modifying them.

In the earlier chapters, we discussed that many real-time security protocols have evolved for network security ensuring basic tenets of security such as privacy, origin authentication, message integrity, and non-repudiation.

Most of these protocols remained focused at the higher layers of the OSI protocol stack, to compensate for inherent lack of security in standard Internet Protocol. Though valuable, these methods cannot be generalized easily for use with any application. For example, SSL is developed specifically to secure applications like HTTP or FTP. But there are several other applications which also need secure communications.

This need gave rise to develop a security solution at the IP layer so that all higher-layer protocols could take advantage of it. In 1992, the Internet Engineering Task Force (IETF) began to define a standard 'IPsec'.

In this chapter, we will discuss how security is achieved at network layer using this very popular set of protocol IPsec.

Security in Network Layer

Any scheme that is developed for providing network security needs to be implemented at some layer in protocol stack as depicted in the diagram below −

Layer	Communication Protocols	Security Protocols
Application Layer	HTTP FTP SMTP	PGP. S/MIME, HTTPS
Transport Layer	TCP /UDP	SSL, TLS, SSH
Network Layer	IP	IPsec

The popular framework developed for ensuring security at network layer is Internet Protocol Security (IPsec).

Features of IPsec

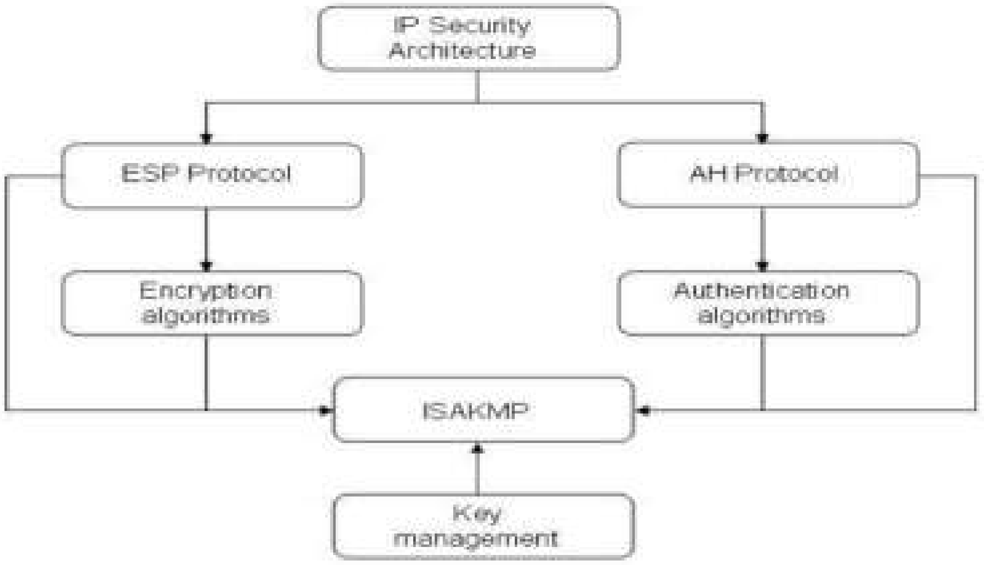
- IPsec is not designed to work only with TCP as a transport protocol. It works with UDP as well as any other protocol above IP such as ICMP, OSPF etc.
- IPsec protects the entire packet presented to IP layer including higher layer headers.
- Since higher layer headers are hidden which carry port number, traffic analysis is more difficult.
- IPsec works from one network entity to another network entity, not from application process to application process. Hence, security can be adopted without requiring changes to individual user computers/applications.
- Tough widely used to provide secure communication between network entities, IPsec can provide host-to-host security as well.
- The most common use of IPsec is to provide a Virtual Private Network (VPN), either between two locations (gateway-to-gateway) or between a remote user and an enterprise network (host-to-gateway).

Security Functions

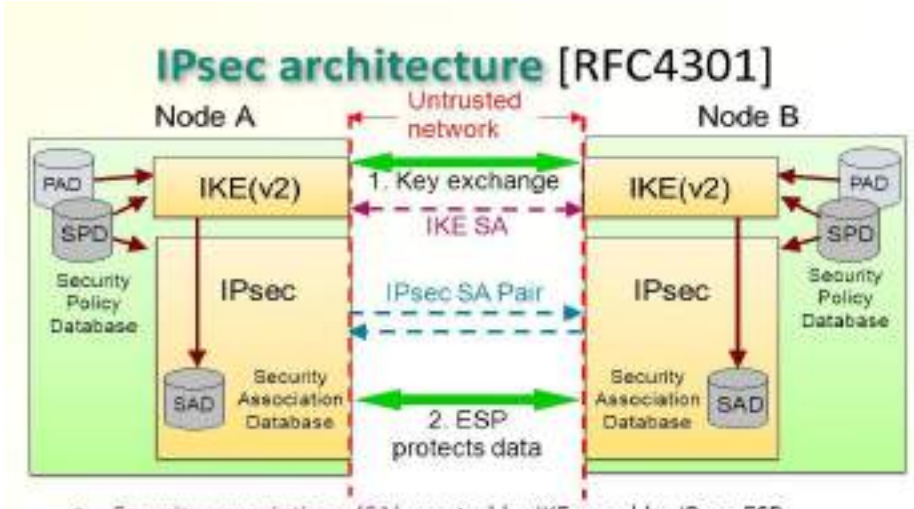
The important security functions provided by the IPsec are as follows −

- Confidentiality
    - Enables communicating nodes to encrypt messages.
    - Prevents eavesdropping by third parties.
- Origin authentication and data integrity.
    - Provides assurance that a received packet was actually transmitted by the party identified as the source in the packet header.
    - Confirms that the packet has not been altered or otherwise.
- Key management.
    - Allows secure exchange of keys.
    - Protection against certain types of security attacks, such as replay attacks.

IPSec Architecture:



IPSec is a suite of three transport-level protocols used for authenticating the origin and content of IP packets and, optionally, for the encryption of their data payload. Two of the protocols, AH (Authentication Header) and ESP (Encapsulating Security Payload) , provide authentication, comprising proof of data source, data integrity and anti-replay protection. Additionally, ESP (but not AH) provides data encryption. The third IPSec protocol, IKE (Internet Key Exchange), is a complex hybrid protocol used for the peer authentication and key exchange processes that necessarily precede the services provided by AH and ESP.



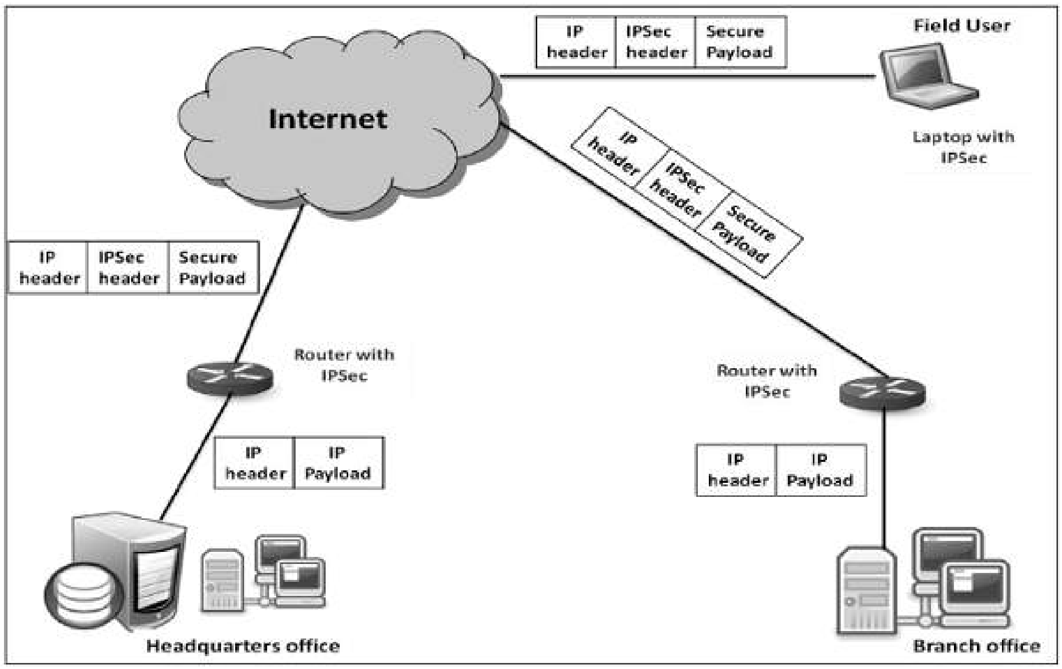
- Security associations (SA) created by IKE, used by IPsec ESP
- Security policy guides SA creation and selection for use
- IPsec is part of the IP layer in the OS kernel; IKE is a user-space service (daemon)

The IPSec protocols do not define *which* algorithms should be used for the computations involved in encryption or in generating digital signatures. This renders the protocol definitions completely generic, meaning they can accommodate developments such as new cryptographic techniques as they become available. The algorithms to be used are specified separately as part of the overall security policy configured at each of the peer stations. The initial IKE negotiations allow the peers to agree on the particular combination of protocols and algorithms to be used for all subsequent IPSec processing.

Virtual Private Network

Ideally, any institution would want its own private network for communication to ensure security. However, it may be very costly to establish and maintain such private network over geographically dispersed area. It would require to manage complex infrastructure of communication links, routers, DNS, etc.

IPsec provides an easy mechanism for implementing Virtual Private Network (VPN) for such institutions. VPN technology allows institution's inter-office traffic to be sent over public Internet by encrypting traffic before entering the public Internet and logically separating it from other traffic. The simplified working of VPN is shown in the following diagram –



Overview of IPsec

IPsec is a framework/suite of protocols for providing security at the IP layer.

Origin

In early 1990s, Internet was used by few institutions, mostly for academic purposes. But in later decades, the growth of Internet became exponential due to expansion of network and several organizations using it for communication and other purposes.

With the massive growth of Internet, combined with the inherent security weaknesses of the TCP/IP protocol, the need was felt for a technology that can provide network security on the Internet. A report entitled "Security in the Internet Architecture" was issued by the Internet Architecture Board (IAB) in 1994. It identified the key areas for security mechanisms.

The IAB included authentication and encryption as essential security features in the IPv6, the next-generation IP. Fortunately, these security capabilities were defined such that they can be implemented with both the current IPv4 and futuristic IPv6.

Security framework, IPsec has been defined in several 'Requests for comments' (RFCs). Some RFCs specify some portions of the protocol, while others address the solution as a whole.

Operations Within IPsec

The IPsec suite can be considered to have two separate operations, when performed in unison, providing a complete set of security services. These two operations are IPsec Communication and Internet Key Exchange.

- IPsec Communication
    - It is typically associated with standard IPsec functionality. It involves encapsulation, encryption, and hashing the IP datagrams and handling all packet processes.
    - It is responsible for managing the communication according to the available Security Associations (SAs) established between communicating parties.
    - It uses security protocols such as Authentication Header (AH) and Encapsulated SP (ESP).
    - IPsec communication is not involved in the creation of keys or their management.

    - IPsec communication operation itself is commonly referred to as IPsec.

- Internet Key Exchange (IKE)
    - IKE is the automatic key management protocol used for IPsec.
    - Technically, key management is not essential for IPsec communication and the keys can be manually managed. However, manual key management is not desirable for large networks.
    - IKE is responsible for creation of keys for IPsec and providing authentication during key establishment process. Though, IPsec can be used for any other key management protocols, IKE is used by default.
    - IKE defines two protocol (Oakley and SKEME) to be used with already defined key management framework Internet Security Association Key Management Protocol (ISAKMP).
    - ISAKMP is not IPsec specific, but provides the framework for creating SAs for any protocol.

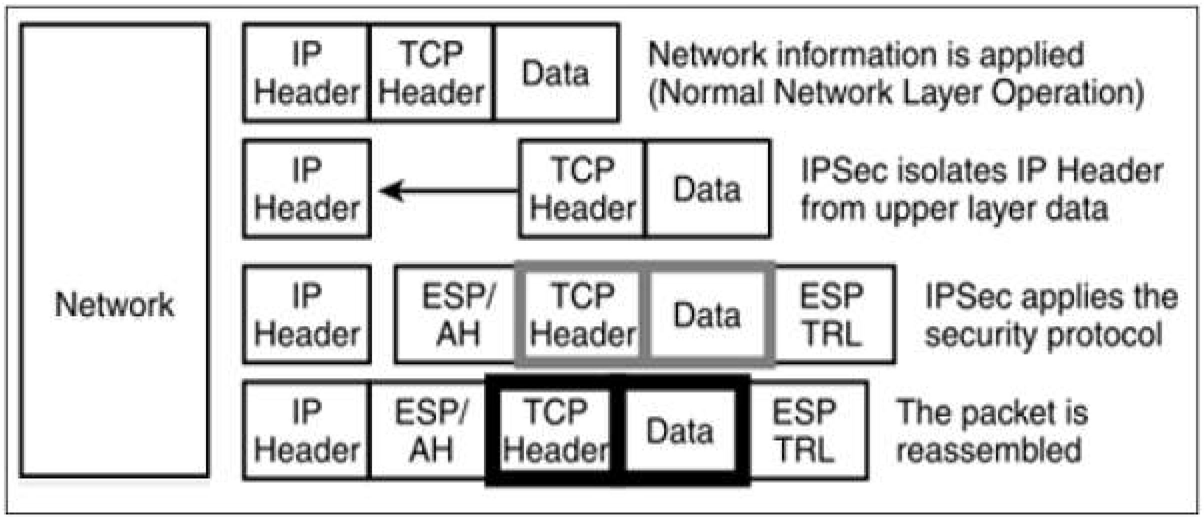
This chapter mainly discusses the IPsec communication and associated protocol employed to achieve security.

IPsec Communication Modes

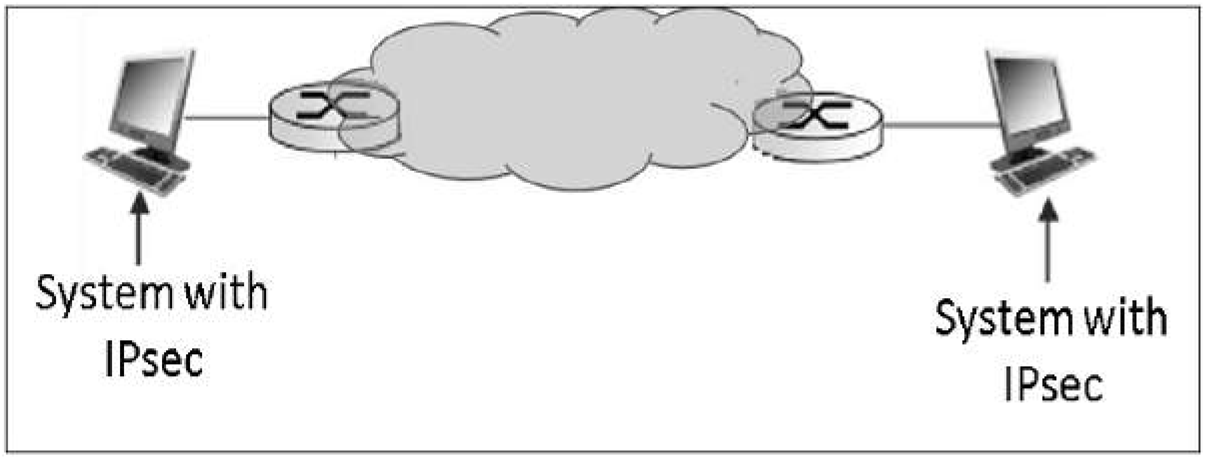
IPsec Communication has two modes of functioning; transport and tunnel modes. These modes can be used in combination or used individually depending upon the type of communication desired.

Transport Mode

- IPsec does not encapsulate a packet received from upper layer.
- The original IP header is maintained and the data is forwarded based on the original attributes set by the upper layer protocol.
- The following diagram shows the data flow in the protocol stack.

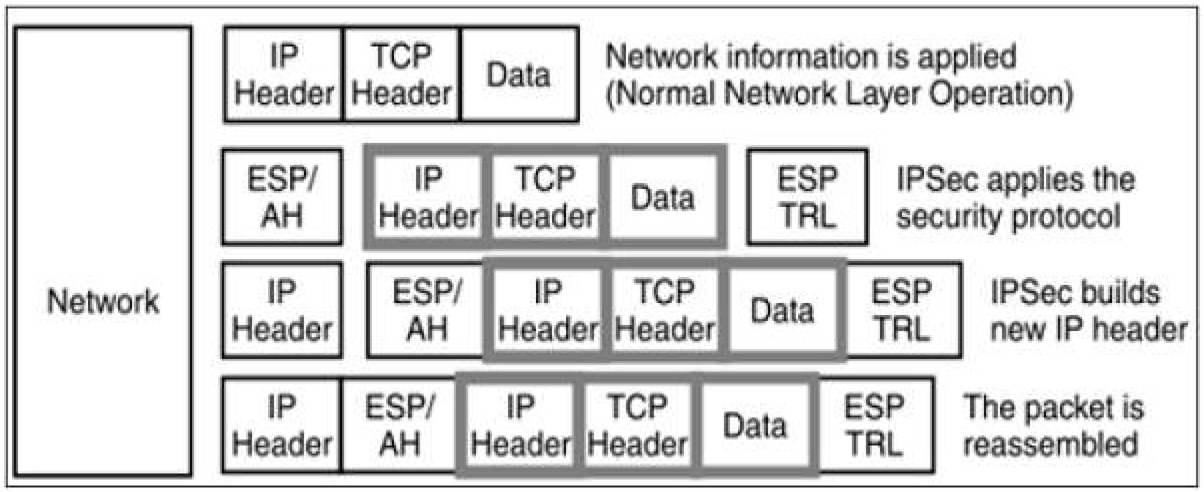

Network
IP Header
TCP Header
Data
Network information is applied (Normal Network Layer Operation)
IP Header
TCP Header
Data
IPSec isolates IP Header from upper layer data
IP Header
ESP/ AH
TCP Header
Data
ESP TRL
IPSec applies the security protocol
IP Header
ESP/ AH
TCP Header
Data
ESP TRL
The packet is reassembled

- The limitation of transport mode is that no gateway services can be provided. It is reserved for point-to-point communications as depicted in the following image.

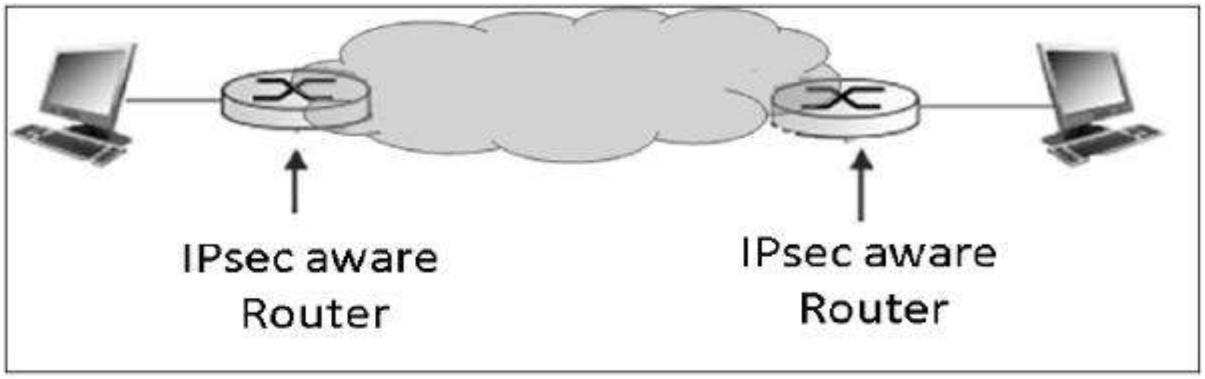


Tunnel Mode

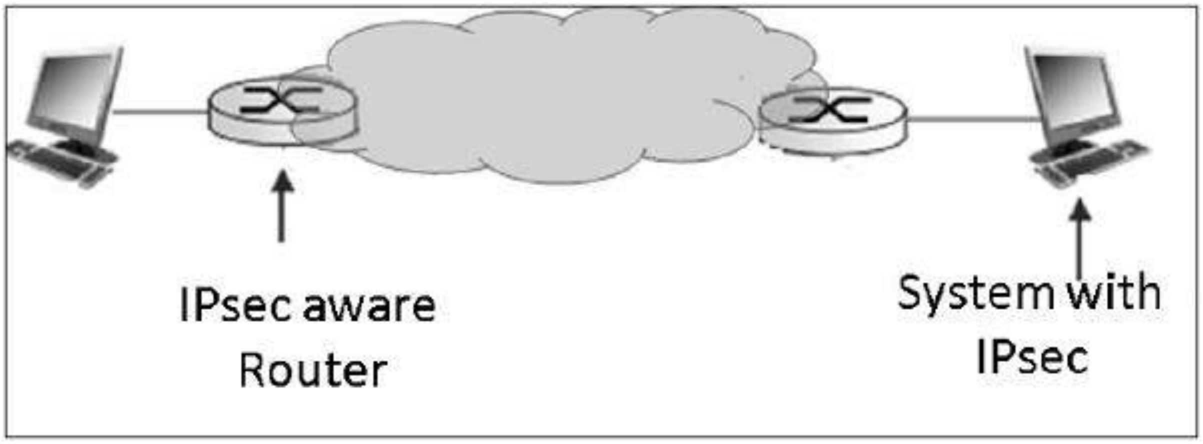
- This mode of IPsec provides encapsulation services along with other security services.
- In tunnel mode operations, the entire packet from upper layer is encapsulated before applying security protocol. New IP header is added.
- The following diagram shows the data flow in the protocol stack.


Network
IP Header
TCP Header
Data
Network information is applied (Normal Network Layer Operation)
ESP/ AH
ESP TRL
IPSec applies the security protocol
IPSec builds new IP header
The packet is reassembled

- Tunnel mode is typically associated with gateway activities. The encapsulation provides the ability to send several sessions through a single gateway.
- The typical tunnel mode communication is as depicted in the following diagram.



- As far as the endpoints are concerned, they have a direct transport layer connection. The datagram from one system forwarded to the gateway is encapsulated and then forwarded to the remote gateway. The remote associated gateway de-encapsulates the data and forwards it to the destination endpoint on the internal network.
- Using IPsec, the tunneling mode can be established between the gateway and individual end system as well.



IPsec Protocols

IPsec uses the security protocols to provide desired security services. These protocols are the heart of IPsec operations and everything else is designed to support these protocol in IPsec.

Security associations between the communicating entities are established and maintained by the security protocol used.

There are two security protocols defined by IPsec — Authentication Header (AH) and Encapsulating Security Payload (ESP).

Authentication Header

The AH protocol provides service of data integrity and origin authentication. It optionally caters for message replay resistance. However, it does not provide any form of confidentiality.

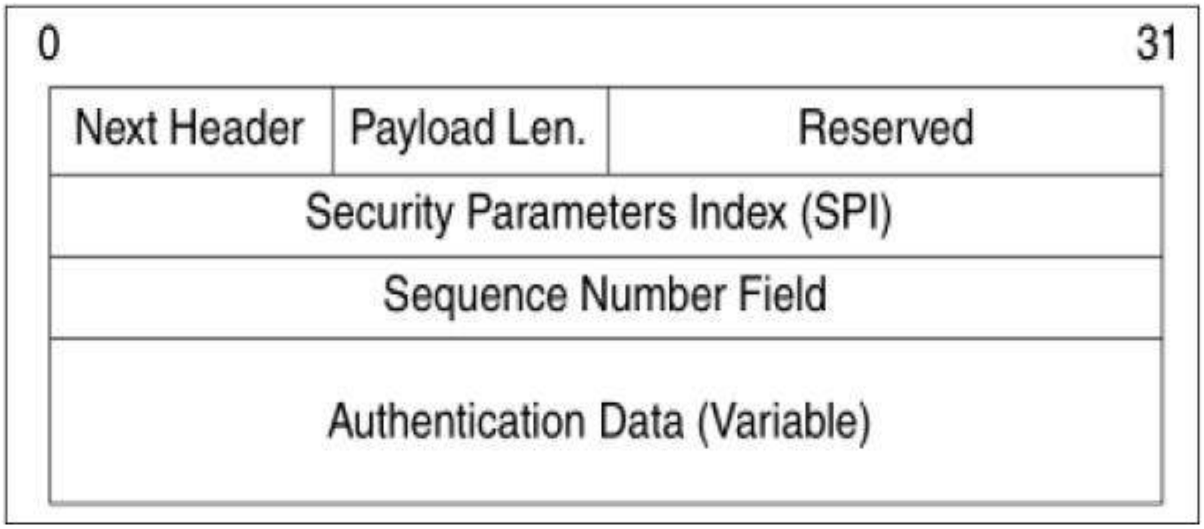
AH is a protocol that provides authentication of either all or part of the contents of a datagram by the addition of a header. The header is calculated based on the values in the datagram. What parts of the datagram are used for the calculation, and where to place the header, depends on the mode cooperation (tunnel or transport).

The operation of the AH protocol is surprisingly simple. It can be considered similar to the algorithms used to calculate checksums or perform CRC checks for error detection.

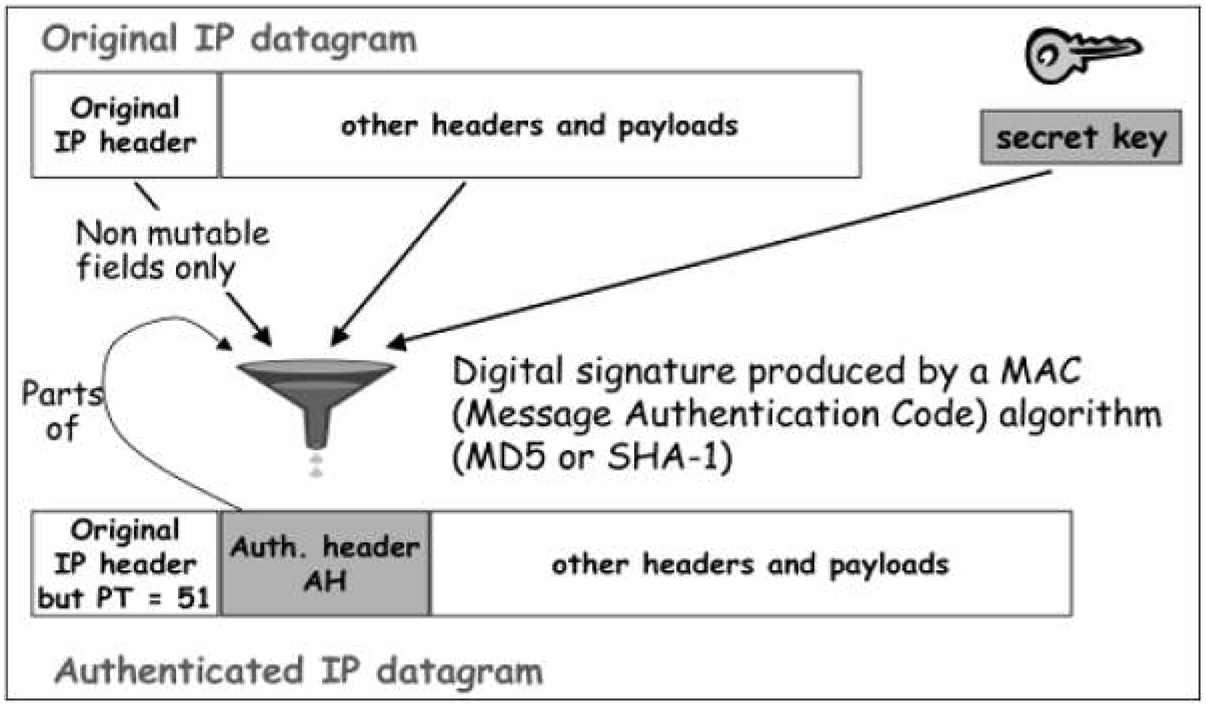
The concept behind AH is the same, except that instead of using a simple algorithm, AH uses special hashing algorithm and a secret key known only to the communicating parties. A security association between two devices is set up that specifies these particulars.

The process of AH goes through the following phases.

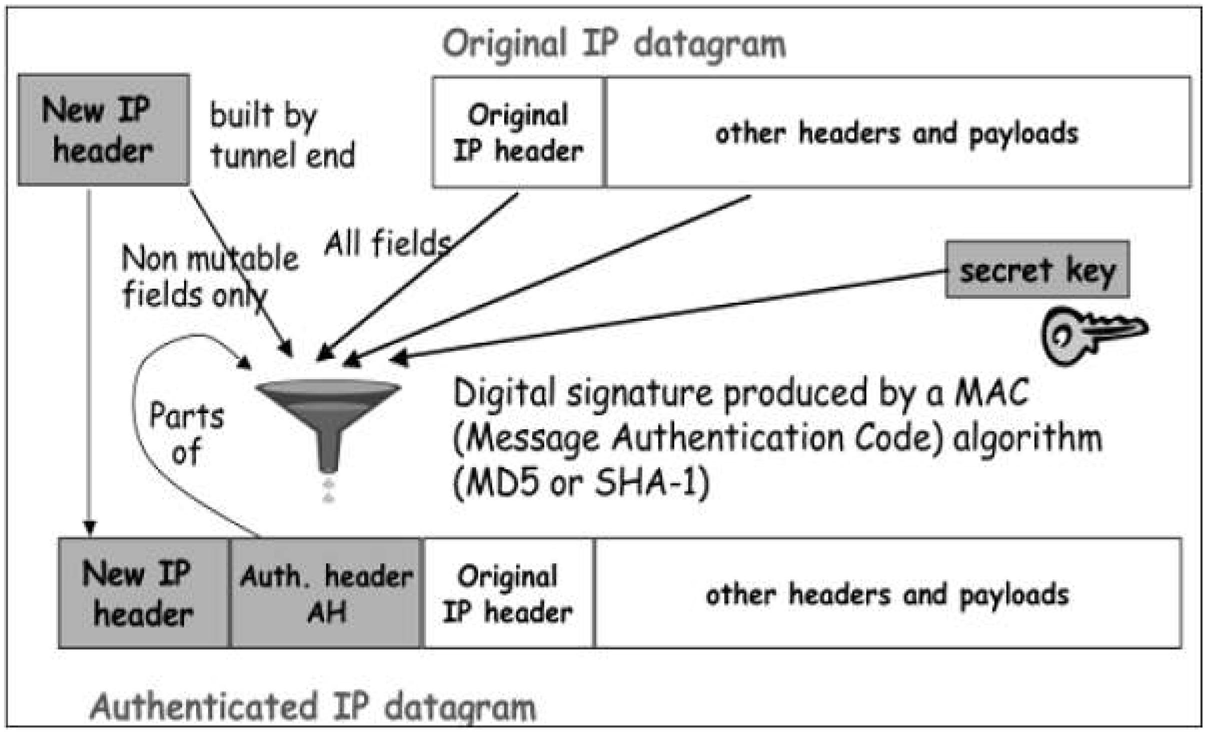
- When IP packet is received from upper protocol stack, IPsec determine the associated Security Association (SA) from available information in the packet; for example, IP address (source and destination).
- From SA, once it is identified that security protocol is AH, the parameters of AH header are calculated. The AH header consists of the following parameters −


0
31
Next Header
Payload Len.
Reserved
Security Parameters Index (SPI)
Sequence Number Field
Authentication Data (Variable)

- The header field specifies the protocol of packet following AH header. Sequence Parameter Index (SPI) is obtained from SA existing between communicating parties.
- Sequence Number is calculated and inserted. These numbers provide optional capability to AH to resist replay attack.
- Authentication data is calculated differently depending upon the communication mode.
- In transport mode, the calculation of authentication data and assembling of final IP packet for transmission is depicted in the following diagram. In original IP header, change is made only in protocol number as 51 to indicated application of AH.


Original IP datagram
Original IP header
other headers and payloads
secret key
Non mutable fields only
Parts of
Digital signature produced by a MAC (Message Authentication Code) algorithm (MD5 or SHA-1)
Original IP header but PT = 51
Auth. header AH
other headers and payloads
Authenticated IP datagram

- In Tunnel mode, the above process takes place as depicted in the following diagram.


Original IP datagram
New IP header
built by tunnel end
Original IP header
other headers and payloads
Non mutable fields only
All fields
secret key
Parts of
Digital signature produced by a MAC (Message Authentication Code) algorithm (MD5 or SHA-1)
New IP header
Auth. header AH
Original IP header
other headers and payloads
Authenticated IP datagram

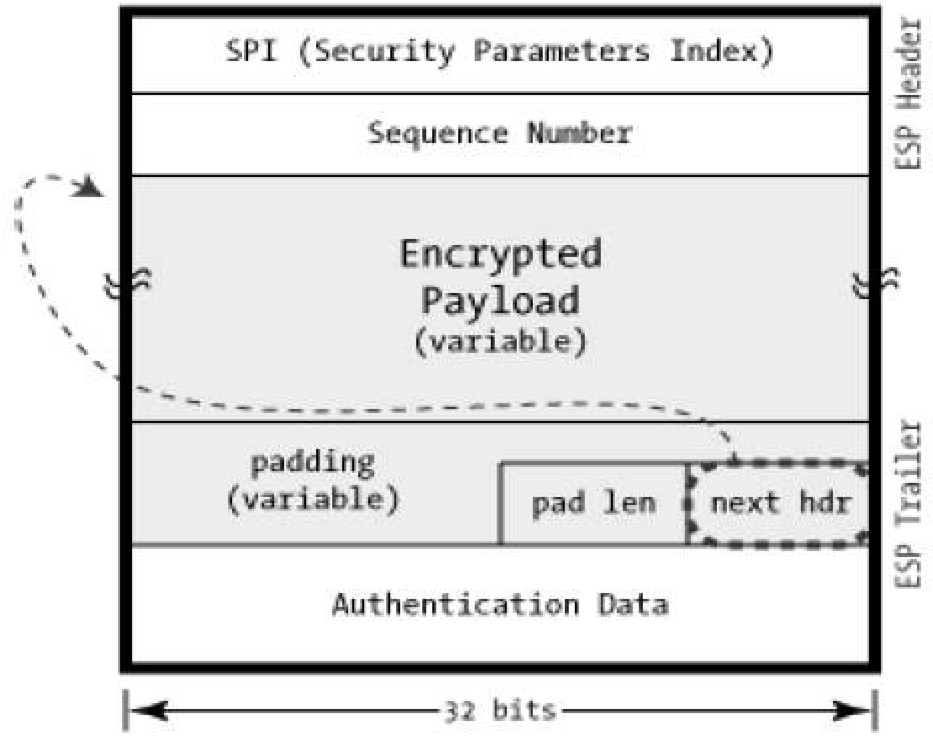
Encapsulation Security Protocol (ESP)

ESP provides security services such as confidentiality, integrity, origin authentication, and optional replay resistance. The set of services provided depends on options selected at the time of Security Association (SA) establishment.

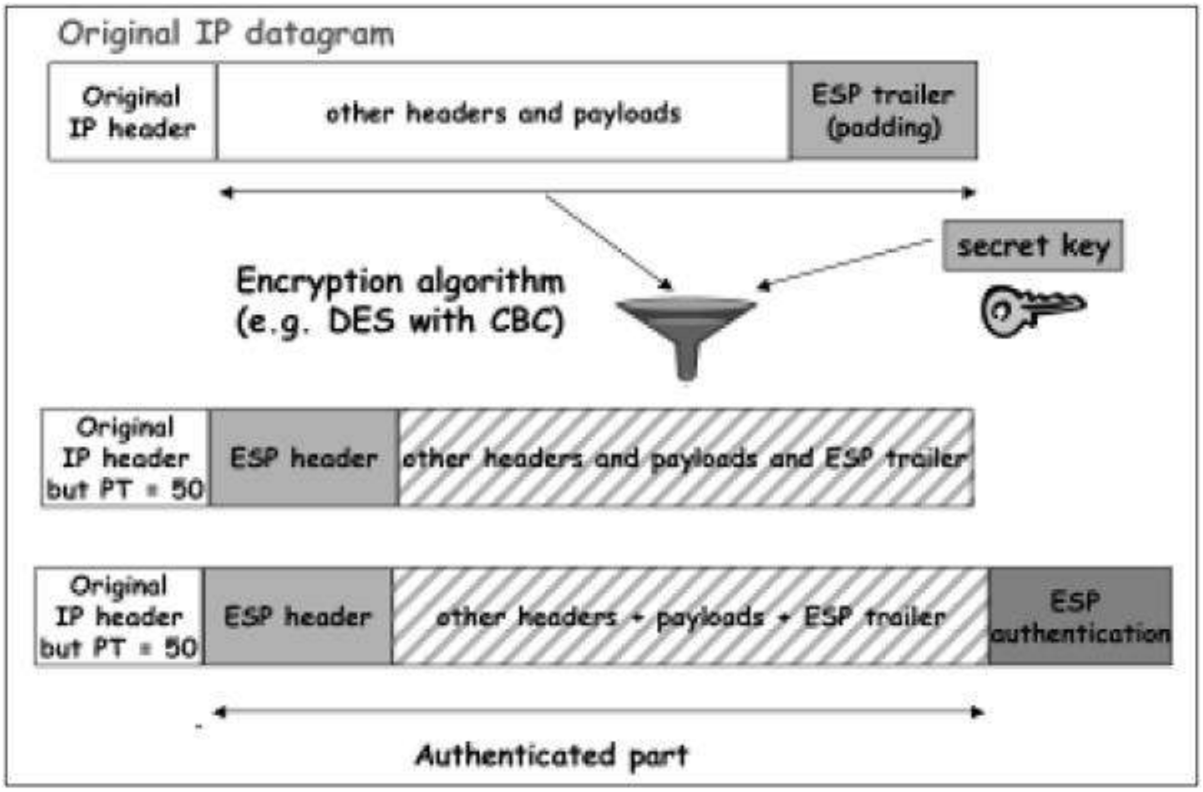
In ESP, algorithms used for encryption and generating authenticator are determined by the attributes used to create the SA.

The process of ESP is as follows. The first two steps are similar to process of AH as stated above.

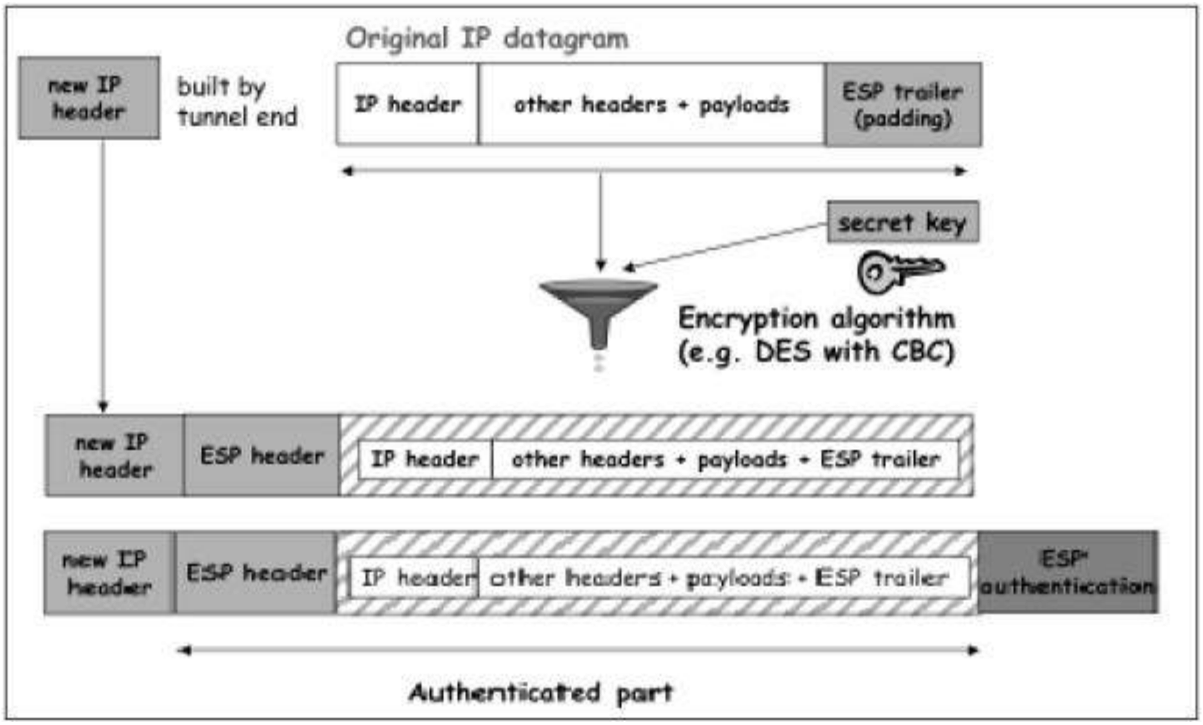
- Once it is determined that ESP is involved, the fields of ESP packet are calculated. The ESP field arrangement is depicted in the following diagram.



- Encryption and authentication process in transport mode is depicted in the following diagram.


Original IP datagram
Original IP header
other headers and payloads
ESP trailer (padding)
secret key
Encryption algorithm (e.g. DES with CBC)
Original IP header but PT = 50
ESP header
other headers and payloads and ESP trailer
Original IP header but PT = 50
ESP header
other headers + payloads + ESP trailer
ESP authentication
Authenticated part

- In case of Tunnel mode, the encryption and authentication process is as depicted in the following diagram.



Although authentication and confidentiality are the primary services provided by ESP, both are optional. Technically, we can use NULL encryption without authentication. However, in practice, one of the two must be implemented to use ESP effectively.

The basic concept is to use ESP when one wants authentication and encryption, and to use AH when one wants extended authentication without encryption.

Security Associations in IPsec

Security Association (SA) is the foundation of an IPsec communication. The features of SA are −

- Before sending data, a virtual connection is established between the sending entity and the receiving entity, called "Security Association (SA)".
- IPsec provides many options for performing network encryption and authentication. Each IPsec connection can provide encryption, integrity, authenticity, or all three services. When the security service is determined, the two IPsec peer entities must determine exactly which algorithms to use (for example, DES or 3DES for encryption; MD5 or SHA-1 for integrity). After deciding on the algorithms, the two devices must share session keys.
- SA is a set of above communication parameters that provides a relationship between two or more systems to build an IPsec session.
- SA is simple in nature and hence two SAs are required for bi-directional communications.
- SAs are identified by a Security Parameter Index (SPI) number that exists in the security protocol header.
- Both sending and receiving entities maintain state information about the SA. It is similar to TCP endpoints which also maintain state information. IPsec is connection-oriented like TCP.

Parameters of SA

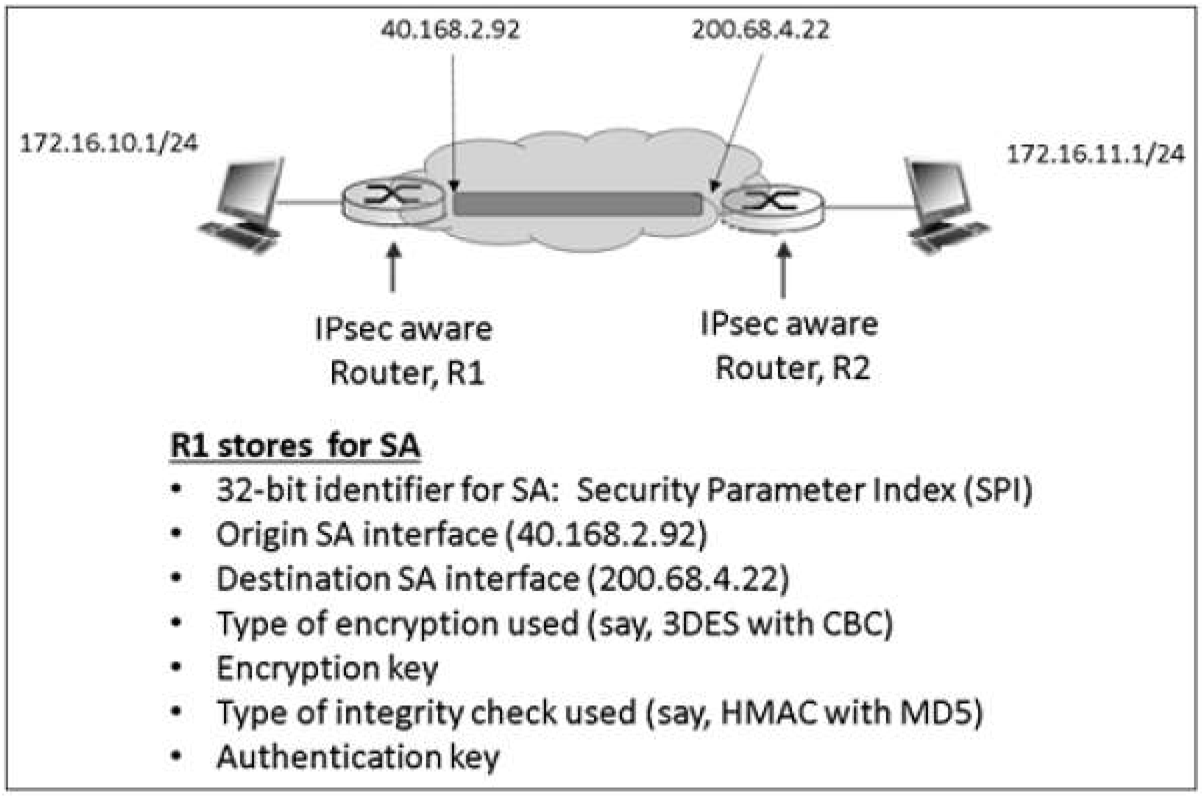
Any SA is uniquely identified by the following three parameters −

- Security Parameters Index (SPI).
    - It is a 32-bit value assigned to SA. It is used to distinguish among different SAs terminating at the same destination and using the same IPsec protocol.

- Every packet of IPsec carries a header containing SPI field. The SPI is provided to map the incoming packet to an SA.
- The SPI is a random number generated by the sender to identify the SA to the recipient.

- **Destination IP Address** – It can be IP address of end router.
- **Security Protocol Identifier** – It indicates whether the association is an AH or ESP SA.

Example of SA between two router involved in IPsec communication is shown in the following diagram.



R1 stores for SA

- 32-bit identifier for SA: Security Parameter Index (SPI)
- Origin SA interface (40.168.2.92)
- Destination SA interface (200.68.4.22)
- Type of encryption used (say, 3DES with CBC)
- Encryption key
- Type of integrity check used (say, HMAC with MD5)
- Authentication key

Security Administrative Databases

In IPsec, there are two databases that control the processing of IPsec datagram. One is the Security Association Database (SAD) and the other is the Security Policy Database (SPD). Each communicating endpoint using IPsec should have a logically separate SAD and SPD.

Security Association Database

In IPsec communication, endpoint holds SA state in Security Association Database (SAD). Each SA entry in SAD database contains nine parameters as shown in the following table −

Sr.No.	Parameters & Description
1	**Sequence Number Counter** For outbound communications. This is the 32-bit sequence number provided in the AH or ESP headers.
2	**Sequence Number Overflow Counter** Sets an option flag to prevent further communications utilizing the specific SA
3	**32-bit anti-replay window** Used to determine whether an inbound AH or ESP packet is a replay
4	**Lifetime of the SA**: Time till SA remain active
5	**Algorithm – AH**: Used in the AH and the associated key
6	**Algorithm - ESP Auth**: Used in the authenticating portion of the ESP header
7	**Algorithm - ESP Encryption** Used in the encryption of the ESP and its associated key information
8	**IPsec mode of operation**: Transport or tunnel mode
9	**Path MTU(PMTU)** Any observed path maximum transmission unit (to avoid fragmentation)

All SA entries in the SAD are indexed by the three SA parameters: Destination IP address, Security Protocol Identifier, and SPI.

Security Policy Database

SPD is used for processing outgoing packets. It helps in deciding what SAD entries should be used. If no SAD entry exists, SPD is used to create new ones.

Any SPD entry would contain −

- Pointer to active SA held in SAD.
- Selector fields – Field in incoming packet from upper layer used to decide application of IPsec. Selectors can include source and destination address, port numbers if relevant, application IDs, protocols, etc.

Outgoing IP datagrams go from the SPD entry to the specific SA, to get encoding parameters. Incoming IPsec datagram get to the correct SA directly using the SPI/DEST IP/Protocol triple, and from there extracts the associated SAD entry.

SPD can also specify traffic that should bypass IPsec. SPD can be considered as a packet filter where the actions decided upon are the activation of SA processes.

Key Management

A security association contains the following information:

- Material for keys for encryption and authentication
- The algorithms that can be used
- The identities of the endpoints
- Other parameters that are used by the system

SAs require keying material for authentication and encryption. The managing of keying material that SAs require is called key management. The Internet Key Exchange (IKE) protocol handles key management automatically. You can also manage keys manually with the ipseckeycommand. SAs on IPv4 and IPv6 packets can use automatic key management.

Keying Utilities

The IKE protocol is the automatic keying utility for IPv4 and IPv6 addresses.The manual keying utility is the ipseckey command.

You use the ipseckey command to manually manipulate the security association databases with the ipsecah and ipsecesp protection mechanisms. You can also use the ipseckey command to set up security associations between communicating parties when automated key management is not used.

Example and usage of some of the ipsec commands are as follows

Command	**Description**
crypto map (global IPSec)	Creates or modifies a crypto map entry and enters the crypto map configuration mode.
crypto map (interface IPSec)	Applies a previously defined crypto map set to an interface.
crypto map local-address	Specifies and names an identifying interface to be used by the crypto map for IPSec traffic.
match address (IPSec)	Specifies an extended access list for a crypto map entry.
set peer (IPSec)	Specifies an IPSec peer in a crypto map entry.
set pfs	Specifies that IPSec should ask for perfect forward secrecy (PFS) when requesting new security associations for this crypto map entry, or that IPSec requires PFS when receiving requests for new security associations.
set security-association lifetime	Overrides (for a particular crypto map entry) the global lifetime value, which is used when negotiating IPSec security associations.
set transform-set	Specifies which transform sets can be used with the crypto map entry.
show crypto dynamic-map	Displays a dynamic crypto map set.
show crypto map (IPSec)	Displays the crypto map configuration.

While the ipseckey command has only a limited number of general options, the command supports a rich command language. You can specify that requests should be delivered by means of a programmatic interface specific for manual keying. See the pf_key(7P) man page for additional information. When you invoke the ipseckey command with no arguments, the command enters an interactive mode that displays a prompt that enables you to make entries. Some commands require an explicit security association (SA) type, while others permit you to specify the SA type and act on all SA types.

Security Considerations for `ipseckey`

The ipseckey command enables a privileged user to enter sensitive cryptographic keying information. If an adversary gains access to this information, the adversary can compromise the security of IPsec traffic. You should consider the following issues when you handle keying material and use the ipseckey command:

1. Have you refreshed the keying material? Periodic key refreshment is a fundamental security practice. Key refreshment guards against potential weaknesses of the algorithm and keys, and limits the damage of an exposed key.
2. Is the TTY going over a network? Is the ipseckey command in interactive mode?
   - In interactive mode, the security of the keying material is the security of the network path for this TTY's traffic. You should avoid using the ipseckey command over a clear-text telnet or rlogin session.
   - Even local windows might be vulnerable to attacks by a concealed program that reads window events.
3. Is the file being accessed over the network? Can the file be read by the world? Have you used the -f option?
   - An adversary can read a network-mounted file as the file is being read. You should avoid using a world-readable file that contains keying material.
   - Protect your naming system. If the following two conditions are met, then your host names are no longer trustworthy:
     - Your source address is a host that can be looked up over the network
     - Your naming system is compromised

Security weaknesses often lie in misapplication of tools, not the actual tools. You should be cautious when using the ipseckey command. Use a console or other hard-connected TTY for the safest mode of operation.

Chapter 5: Computer Security

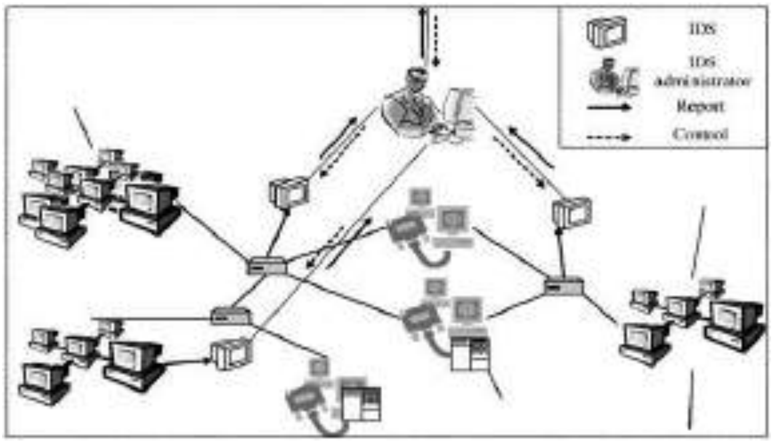
Intrusion detection system

An intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system.

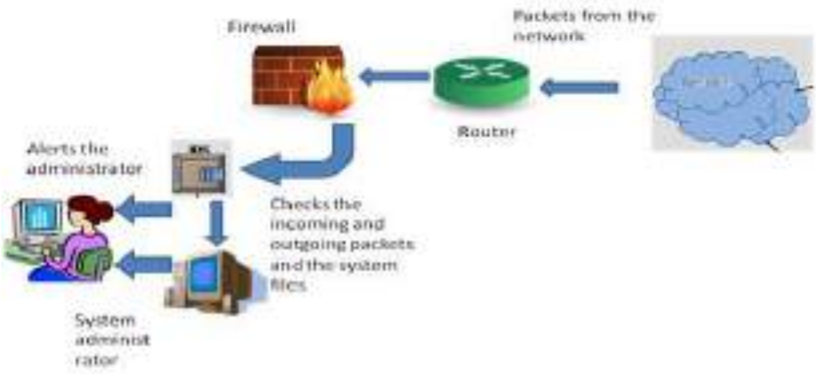
Classification of Intrusion Detection System:

Based on the type of systems the IDS protects:

- **Network Intrusion Detection System**: This system monitors the traffic on individual networks or subnets by continuously analyzing the traffic and comparing it with the known attacks in the library. If an attack is detected, an alert is sent to the system administration. It is placed mostly at important points in the network so that it can keep an eye on the traffic travelling to and from the different devices on the network. The IDS is placed along the network boundary or between the network and the server. An advantage of this system is that it can be deployed easily and at low cost, without having to be loaded for each system.

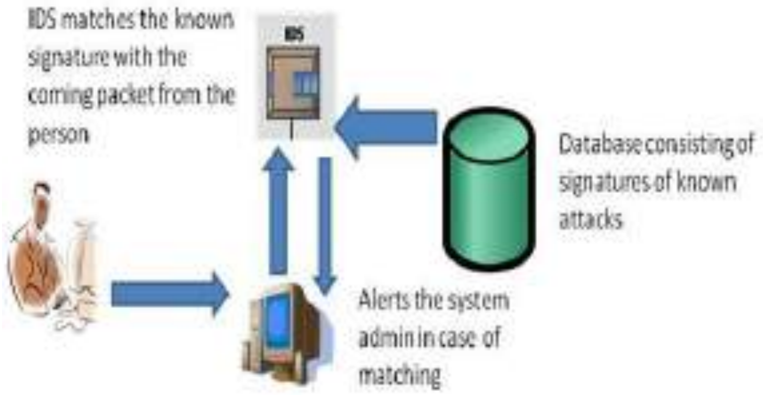


- **Host Intrusion Detection System**: Such system works on individual systems where the network connection to the system, i.e. incoming and outgoing of packets are constantly monitored and also the auditing of system files is done and in case of any discrepancy, the system administrator is alerted about the same. This system monitors the operating system of the computer. The IDS is installed on the computer. Advantage of this system is it can accurately monitor the whole system and does not require installation of any other hardware.

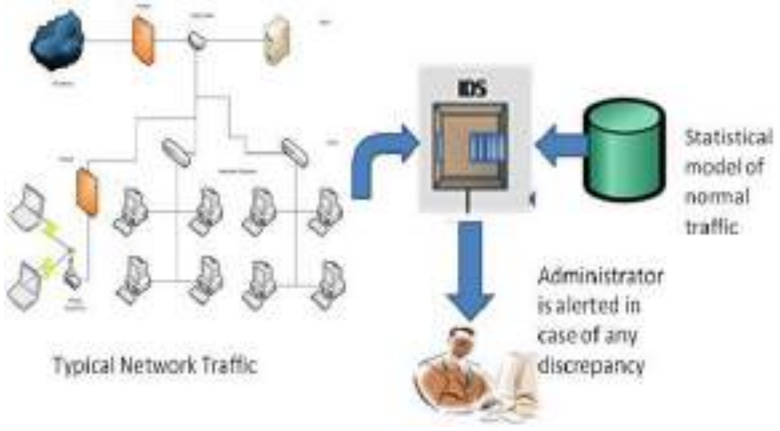


Based on the method of working:

- **Signature based Intrusion Detection System**: This system works on the principle of matching. The data is analyzed and compared with the signature of known attacks. Incase of any matching, an alert is issued. An advantage of this system is it has more accuracy and standard alarms understood by user.

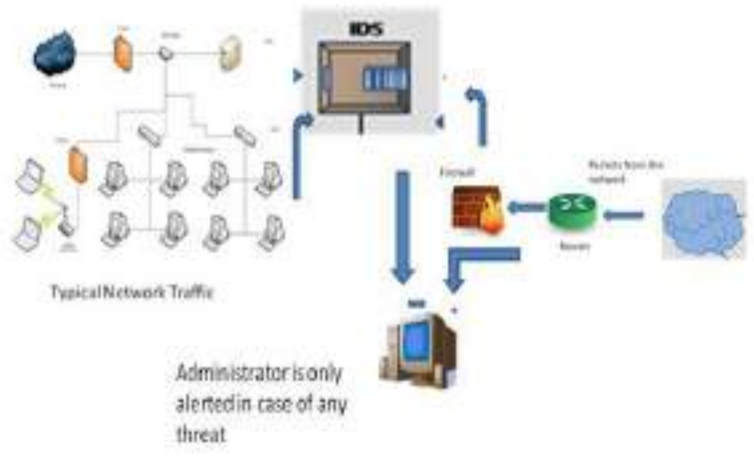

IDS matches the known signature with the coming packet from the person
IDS
Database consisting of signatures of known attacks
Alerts the system admin in case of matching

- **Anomaly based Intrusion Detection System**: It consists of a statistical model of a normal network traffic which consists of the bandwidth used, the protocols defined for the traffic, the ports and devices which are part of the network. It regularly monitors the network traffic and compares it with the statistical model. In case of any anomaly or discrepancy, the administrator is alerted. An advantage of this system is they can detect new and unique attacks.


IDS
Statistical model of normal traffic
Administrator is alerted in case of any discrepancy
Typical Network Traffic

Based on their Functioning:

- **Passive Intrusion Detection System**: It simply detects the kind of malware operation and issues an alert to the system or network administrator. (What we have been seeing till now!).The required action is then taken by the administrator.



- **Reactive Intrusion Detection System**: It not only detects the threat but also performs specific action by resetting the suspicious connection or blocks the network traffic from the suspicious source. It is also known as Intrusion Prevention System.

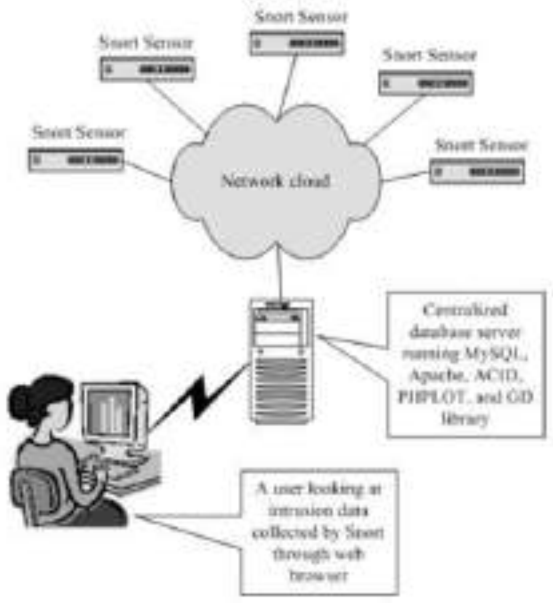
Typical Features of an Intrusion Detection System:

- It monitors and analysis the user and system activities.
- It performs auditing of the system files and other configurations and the operating system.
- It assesses the integrity of system and data files
- It conducts analysis of patterns based on known attacks.
- It detects errors in system configuration.
- It detects and cautions if the system is in danger.

Free Intrusion Detection Software

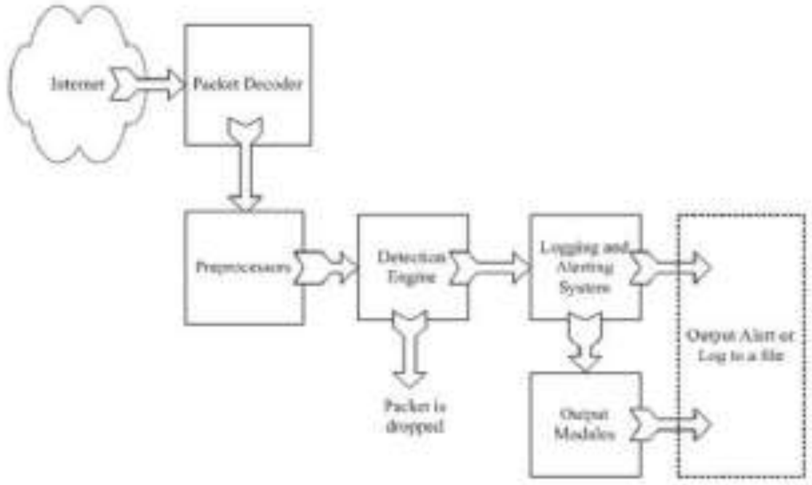
Snort Intrusion Detection System

One of the most widely used Intrusion Detection Software is the Snort software. It is a network Intrusion Detection Software developed by Source file. It performs real time traffic analysis and protocol analysis, pattern matching and detection of various kinds of attacks.


Snort Sensor
Snort Sensor
Snort Sensor
Snort Sensor
Snort Sensor
Network cloud
Centralized database server running MySQL, Apache, ACID, PHPLOT, and GD library
A user looking at intrusion data collected by Snort through web browser

Snort Intrusion Detection System

A Snort based Intrusion Detection System Consists of the following Components:



Components of Snort IDS by Intrusion Detection System with Snort

- **A Packet Decoder**: It takes packets from different networks and prepares them for preprocessing or any further action. It basically decodes the coming network packets.
- **A Preprocessor**: It prepares and modifies the data packets and also perform defragmentation of data packets, decodes the tcp streams.
- **A Detection Engine**: It performs the packet detection on basis of Snort rules. If any packet matches the rules, appropriate action is taken, else it is dropped.
- **Logging and Alerting System**: The detected packet is either logged in system files or incase of threats, the system is alerted.
- **Output Modules**: They control the type of output from the logging and alert system.

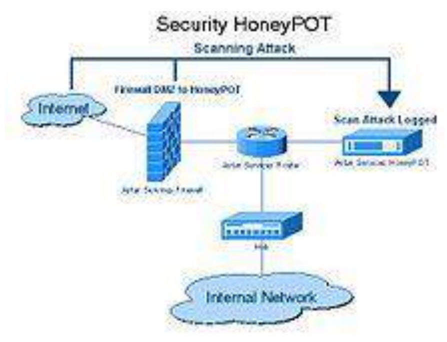
Advantages of Intrusion Detection Systems

- The network or computer is constantly monitored for any invasion or attack.
- The system can be modified and changed according to needs of specific client and can help outside as well as inner threats to the system and network.
- It effectively prevents any damage to the network.
- It provides user friendly interface which allows easy security management systems.
- Any alterations to files and directories on the system can be easily detected and reported.

An only disadvantage of Intrusion Detection System is they cannot detect the source of the attack and in any case of attack, they just lock the whole network.

HONEY POT

In computer terminology, a honeypot is a computer security mechanism set to detect, deflect, or, in some manner, counteract attempts at unauthorized use of information systems. Generally, a honeypot consists of data (for example, in a network site) that appears to be a legitimate part of the site, but is actually isolated and monitored, and that seems to contain information or a resource of value to attackers, who are then blocked. This is similar to police sting operations, colloquially known "baiting," a suspect.



PASSWORD MANAGEMENT

The most widely used method to prevent unauthorized access is to use passwords. A password is a string of characters used to authenticate a user to access a system. The password needs to be kept secret and is only intended for the specific user. In computer systems, each password is associated with a specific username since many individuals may be accessing the same system. Good passwords are essential to keeping computer systems secure. Unfortunately, many computer users don't use very secure passwords, such as the name of a family member or important dates.

Simple tips for constructing a hack-proof password are as follows.

1) Longer is usually stronger. Passwords featuring 10 or more characters are better than those with 8 or less. Try experimenting with login phrases instead of single words

2) Use uppercase and lowercase letters. Try a combination of big and small letters, and in random combination not always initial letter capitalization.

3) Insert numbers and special characters. Substituting a zero “0” for the letter “O” is one common method, but also try 1 for I, 3 for E, and 5 for S. Add characters like @#$%^&* for variety.

4) Experiment with clues. Think of a random childhood attraction, or a place you love, or a specific car, a vacation spot, or a favorite restaurant. These will be easy to remember but hard to crack using what may be already known about you.

5) Use a personal algorithm. You can create your own cryptographic method to obscure your passwords. Try thinking of a long phrase and then using just the initial letters of that phrase. Combine unrelated words. Always substitute the same numbers for certain letters. Type the password one row higher on the keyboard.

6) Change often. Changing your passwords monthly, even occasionally, is a good practice.

ONE-TIME-PASSWORD TOKEN

Another technology that may be used to facilitate password management is the one-time-password token. Users authenticate themselves with two unique factors, something they have (the token) and something they know (the PIN). Users do not need to choose or memories passwords. The token will generate a unique, one-time-use password for each authentication process, based on the PIN and other factors, granting access to protected resources.

1) A token is needed for each user of the authentication process, which implies additional investment.

2)) Users must carry the token at all times, and they will not be able to access the system if they lose the token or forget to bring it with them. Unlike software-based access control systems, which only require a password reset, users may not be able to use the system for hours or days if the token is lost.

3) 3) Users should be aware of the physical security of the token and ensure that the token is properly protected at all times.

4) 4) Most of the current one-time-password authentication schemes only authenticate the initial connection. Connections thereafter are assumed to be authenticated, and these connections are susceptible to being hijacked.

5) Security tokens may not support all applications or servers.

Password length is only part of the problem. Many people, when permitted to choose their own password, pick a password that is guessable, such as their own name, their street name, a common dictionary word, and so forth. This makes the job of password cracking straightforward. The cracker simply has to test the password file against lists of likely passwords. Because many people use guessable passwords, such a strategy should succeed on virtually all systems. In Figure 2 we see that 46% of respondents used passwords with a length equal to or less than 6 characters – even though it means that 54% used passwords of length 7 or more, the statistics are simply not favorably[9]. The length requirement is an even more basic requirement which students either choose to ignore or they are not aware of it. The average length of all passwords appears to be appropriate at between 8 and 9 characters. Almost half of the respondents (49%) stated that they use secure passwords. When checking their passwords and applying only the one rule concerning password length, it was found however, that 46% of those who said that they are using secure passwords have passwords with 6, or less, characters. Password strength is the likelihood that a password cannot be guessed or discovered, and varies with the attack algorithm used.

PASSWORD SELECTION STRATEGIES

If users are assigned passwords consisting of eight randomly selected printable characters, password cracking is effectively impossible. But it would be almost as impossible for most users to remember their passwords. Our goal, then is to eliminate guessable passwords while allowing the user to select a password that is memorable. Four basic techniques are in use:

A. **User Education:** Users can be told the importance of using hard to guess passwords and can be provided with guidelines for selecting strong passwords. This user education strategy is unlikely to succeed at most installations, particularly where there is a large user population or a lot of turnover. Many users will simply ignore the guidelines. Others may not be good judges of what is a strong password.

B. Computer-Generated Passwords: Computer-generated passwords also have problems. If the passwords are quite random in nature, users will not be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down. In general, computer-generated password schemes have a history of poor acceptance by users.

C. Reactive Password Checking: A reactive password checking strategy is one in which the system periodically run its own password cracker to find guessable passwords. The system cancels any passwords that are guessed and notifies the user. This tactic has a number of drawbacks. First, it is resource intensive if the job is done right. Because a determined opponent who is able to steal a password checker is at a distinct disadvantage.

D. Proactive Password Checking: The most promising approach to improved password security is a proactive password checker. In this scheme, a user is allowed to select his or her own password. However, at the time of selection, the system checks to see if the password is allowable and, if not, rejects it. Such checkers are based on the philosophy that, with sufficient guidance from the system, user can select memorable passwords from a fairly large password space that are not likely to be guessed in a dictionary attack.

Viruses and Related Threats

Perhaps the most sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems. In this context, we are concerned with application programs as well as utility programs, such as editors and compilers.

We begin this section with an overview of the spectrum of such software threats. The remainder of the section is devoted to viruses and worms.

Malicious Programs

The terminology in this area presents problems because of a lack of universal agreement on all of the terms and because some of the categories overlap. Table 19.1, based principally on [SZOR05], is a useful guide.

Terminology of Malicious Programs

Name	Description
Virus	Attaches itself to a program and propagates copies of itself to other programs
Worm	Program that propagates copies of itself to other computers
Logic bomb	Triggers action when condition occurs
Trojan horse	Program that contains unexpected additional functionality
Backdoor (trapdoor)	Program modification that allows unauthorized access to functionality
Exploits	Code specific to a single vulnerability or set of vulnerabilities
Downloaders	Program that installs other items on a machine that is under attack. Usually, a downloader is sent in an e-mail.
Auto-rooter	Malicious hacker tools used to break into new machines remotely
Kit (virus generator)	Set of tools for generating new viruses automatically
Spammer programs	Used to send large volumes of unwanted e-mail
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial of service (DoS) attack
Keyloggers	Captures keystrokes on a compromised system
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access
Zombie	Program activated on an infected machine that is activated to launch attacks on other machines

Malicious software can be divided into two categories: those that need a host program, and those that are independent. The former are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples. The latter are self-contained programs that can be scheduled and run by the operating system. Worms and zombie programs are examples.

We can also differentiate between those software threats that do not replicate and those that do. The former are programs or fragments of programs that are activated by a trigger. Examples are logic bombs, backdoors, and zombie programs. The latter consist of either a program fragment or an independent program that, when executed, may produce one or more copies of itself to be activated later on the same system or some other system. Viruses and worms are examples.

Backdoor

A backdoor, also known as a trapdoor, is a secret entry point into a program that allows someone that is aware of the backdoor to gain access without going through the usual security access procedures. Programmers have used backdoors legitimately for many years to debug and test programs. This usually is done when the programmer is developing an application that has an authentication procedure, or a long setup, requiring the user to enter many different values to run the application. To debug the program, the developer may wish to gain special privileges or to avoid all the necessary setup and authentication. The programmer may also want to ensure that there is a method of activating the program should something be wrong with the authentication procedure that is being built into the application. The backdoor is code that recognizes some special sequence of input or is triggered by being run from a certain user ID or by an unlikely sequence of events.

Backdoors become threats when unscrupulous programmers use them to gain unauthorized access. The backdoor was the basic idea for the vulnerability portrayed in the movie War Games. Another example is that during the development of Multics, penetration tests were conducted by an Air Force "tiger team" (simulating adversaries). One tactic employed was to send a bogus operating system update to a site running Multics. The update contained a Trojan horse (described later) that could be activated by a backdoor and that allowed the tiger team to gain access. The threat was so well implemented that the Multics developers could not find it, even after they were informed of its presence.

It is difficult to implement operating system controls for backdoors. Security measures must focus on the program development and software update activities.

Logic Bomb

One of the oldest types of program threat, predating viruses and worms, is the logic bomb. The logic bomb is code embedded in some legitimate program that is set to "explode" when certain conditions are met. Examples of conditions that can be used as triggers for a logic bomb are the presence or absence of certain files, a particular day of the week or date, or a particular user running the application. Once triggered, a bomb may alter or delete data or entire files, cause a machine halt, or do some other damage. A striking example of how logic bombs can be employed was the case of Tim Lloyd, who was convicted of setting a logic bomb that cost his employer, Omega Engineering, more than $10 million, derailed its corporate growth strategy, and eventually led to the layoff of 80 workers [GAUD00]. Ultimately, Lloyd was sentenced to 41 months in prison and ordered to pay $2 million in restitution.

Trojan Horses

A Trojan horse is a useful, or apparently useful, program or command procedure containing hidden code that, when invoked, performs some unwanted or harmful function.

Trojan horse programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly. For example, to gain access to the files of another user on a shared system, a user could create a Trojan horse program that, when executed, changed the invoking user's file permissions so that the files are readable by any user. The author could then induce users to run the program by placing it in a common directory and naming it such that it appears to be a useful utility. An example is a program that ostensibly produces a listing of the user's files in a desirable format. After another user has run the program, the author can then access the information in the user's files. An example of a Trojan horse program that would be difficult to detect is a compiler that has been modified to insert additional code into certain programs as they are compiled, such as a system login program [THOM84]. The code creates a backdoor in the login program that permits the author to log on to the system using a special password. This Trojan horse can never be discovered by reading the source code of the login program.

Another common motivation for the Trojan horse is data destruction. The program appears to be performing a useful function (e.g., a calculator program), but it may also be quietly deleting the user's files. For example, a CBS executive was victimized by a Trojan horse that destroyed all information contained in his computer's memory [TIME90]. The Trojan horse was implanted in a graphics routine offered on an electronic bulletin board system.

Zombie

A zombie is a program that secretly takes over another Internet-attached computer and then uses that computer to launch attacks that are difficult to trace to the zombie's creator. Zombies are used in denial-of-service attacks, typically against targeted Web sites. The zombie is planted on hundreds of computers belonging to unsuspecting third parties, and then used to overwhelm the target Web site by launching an overwhelming onslaught of Internet traffic. Section 19.3 discusses zombies in the context of denial of service attacks.

The Nature of Viruses

A virus is a piece of software that can "infect" other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs.

Biological viruses are tiny scraps of genetic codeDNA or RNAthat can take over the machinery of a living cell and trick it into making thousands of flawless replicas of the original virus. Like its biological counterpart, a computer virus carries in its instructional code the recipe for making perfect copies of itself. The typical virus becomes embedded in a program on a computer. Then, whenever the infected computer comes into contact with an uninfected piece of software, a fresh copy of the virus passes into the new program. Thus, the infection can be spread from computer to computer by unsuspecting users who either swap disks or send programs to one another over a network. In a network environment, the ability to access applications and system services on other computers provides a perfect culture for the spread of a virus.

A virus can do anything that other programs do. The only difference is that it attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs.

During its lifetime, a typical virus goes through the following four phases:

- **Dormantphase**: The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.
- **Propagation phase:** The virus places an identical copy of itself into other programs or into certain system areas on the disk. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.
- **Triggering phase:** The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a

variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.

- **Execution phase:** The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

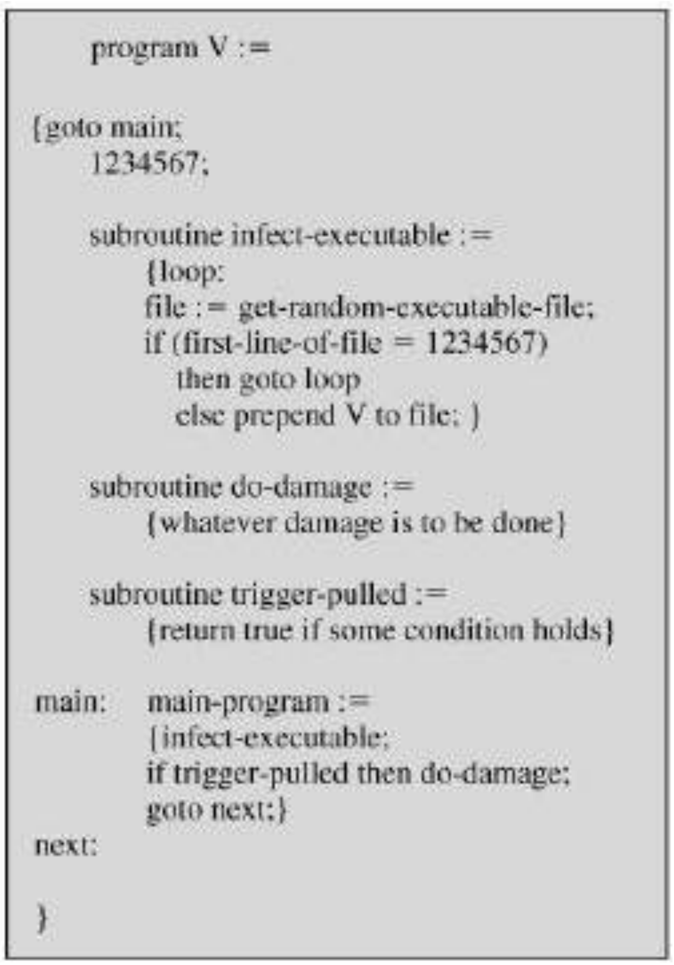
Most viruses carry out their work in a manner that is specific to a particular operating system and, in some cases, specific to a particular hardware platform. Thus, they are designed to take advantage of the details and weaknesses of particular systems.

Virus Structure

A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program.

A very general depiction of virus structure is shown in Figure 19.1 (based on [COHE94]. In this case, the virus code, V, is prepended to infected programs, and it is assumed that the entry point to the program, when invoked, is the first line of the program.

A Simple Virus



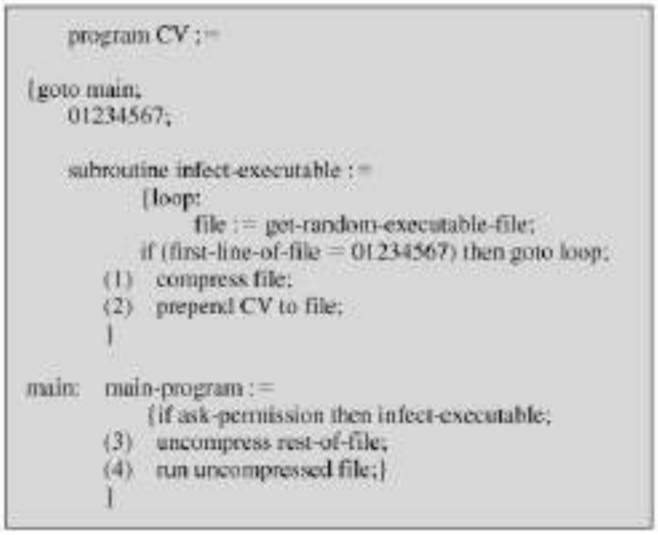
```
    program V :=

{goto main;
    1234567;

    subroutine infect-executable :=
        {loop:
        file := get-random-executable-file;
        if (first-line-of-file = 1234567)
           then goto loop
           else prepend V to file; }

    subroutine do-damage :=
        {whatever damage is to be done}

    subroutine trigger-pulled :=
        {return true if some condition holds}

main:   main-program :=
        {infect-executable;
        if trigger-pulled then do-damage;
        goto next;}
next:

}
```

An infected program begins with the virus code and works as follows. The first line of code is a jump to the main virus program. The second line is a special marker that is used by the virus to determine whether or not a potential victim program has already been infected with this virus. When the program is invoked, control is immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them. Next, the virus may perform some action, usually detrimental to the system. This action could be performed every time the program is invoked, or it could be a logic bomb that triggers only under certain conditions. Finally, the virus transfers control to the original program. If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and uninfected program.

A virus such as the one just described is easily detected because an infected version of a program is longer than the corresponding uninfected one. A way to thwart such a simple means of detecting a virus is to compress the executable file so that both the infected and uninfected versions are of identical length. Figure 19.2 [COHE94] shows in general terms the logic required. The key lines in this virus are numbered, and Figure 19.3 [COHE94] illustrates the operation. We assume that program P_1 is infected with the virus CV. When this program is invoked, control passes to its virus, which performs the following steps:

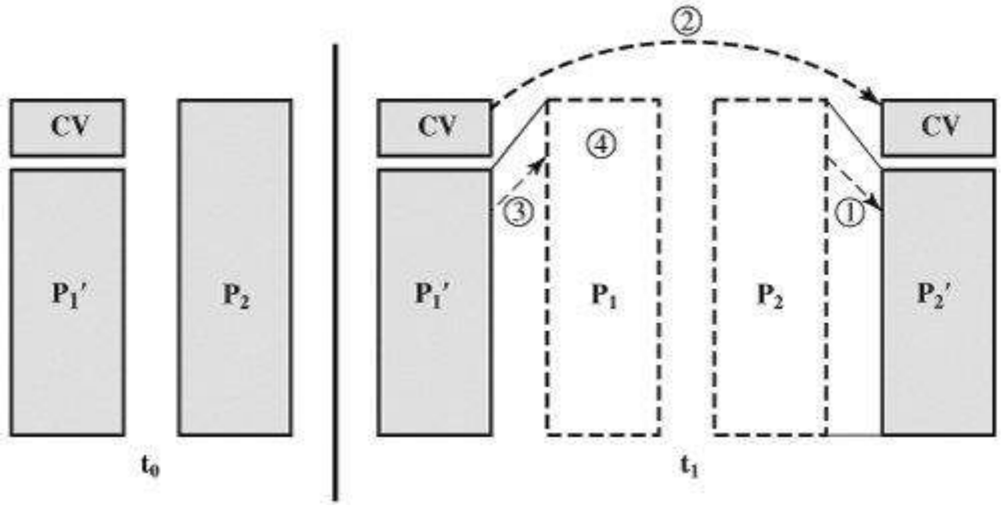
1. For each uninfected file P_2 that is found, the virus first compresses that file to produce P'_2, which is shorter than the original program by the size of the virus.
2. A copy of the virus is prepended to the compressed program.
3. The compressed version of the original infected program, P'_1, is uncompressed.
4. The uncompressed original program is executed.

Logic for a Compression Virus



```
    program CV :=

{goto main;
    01234567;

    subroutine infect-executable :=
            {loop:
                    file := get-random-executable-file;
            if (first-line-of-file = 01234567) then goto loop;
        (1)   compress file;
        (2)   prepend CV to file;
        }

main:   main-program :=
            {if ask-permission then infect-executable;
        (3)   uncompress rest-of-file;
        (4)   run uncompressed file;}
        }
```

A Compression Virus



In this example, the virus does nothing other than propagate. As in the previous example, the virus may include a logic bomb.

Initial Infection

Once a virus has gained entry to a system by infecting a single program, it is in a position to infect some or all other executable files on that system when the infected program executes. Thus, viral infection can be completely prevented by preventing the virus from gaining entry in the first place. Unfortunately, prevention is extraordinarily difficult because a virus can be part of any program outside a system. Thus, unless one is content to take an absolutely bare piece of iron and write all one's own system and application programs, one is vulnerable.

Types of Viruses

There has been a continuous arms race between virus writers and writers of antivirus software since viruses first appeared. As effective countermeasures have been developed for existing types of viruses, new types have been developed. [STEP93] suggests the following categories as being among the most significant types of viruses:

- **Parasitic virus:** The traditional and still most common form of virus. A parasitic virus attaches itself to executable files and replicates, when the infected program is executed, by finding other executable files to infect.
- **Memory-resident virus:** Lodges in main memory as part of a resident system program. From that point on, the virus infects every program that executes.
- **Boot sector virus:** Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.

- **Stealth virus:** A form of virus explicitly designed to hide itself from detection by antivirus software.
- **Polymorphic virus**: A virus that mutates with every infection, making detection by the "signature" of the virus impossible.
- **Metamorphic virus**: As with a polymorphic virus, a metamorphic virus mutates with every infection. The difference is that a metamorphic virus rewrites itself completely at each iteration, increasing the difficulty of detection. Metamorphic viruses my change their behavior as well as their appearance.

One example of a **stealth virus** was discussed earlier: a virus that uses compression so that the infected program is exactly the same length as an uninfected version. Far more sophisticated techniques are possible. For example, a virus can place intercept logic in disk I/O routines, so that when there is an attempt to read suspected portions of the disk using these routines, the virus will present back the original, uninfected program. Thus, stealth is not a term that applies to a virus as such but, rather, is a technique used by a virus to evade detection.

A **polymorphic virus** creates copies during replication that are functionally equivalent but have distinctly different bit patterns. As with a stealth virus, the purpose is to defeat programs that scan for viruses. In this case, the "signature" of the virus will vary with each copy. To achieve this variation, the virus may randomly insert superfluous instructions or interchange the order of independent instructions. A more effective approach is to use encryption. A portion of the virus, generally called a mutation engine, creates a random encryption key to encrypt the remainder of the virus. The key is stored with the virus, and the mutation engine itself is altered. When an infected program is invoked, the virus uses the stored random key to decrypt the virus. When the virus replicates, a different random key is selected.

Another weapon in the virus writers' armory is the virus-creation toolkit. Such a toolkit enables a relative novice to create quickly a number of different viruses. Although viruses created with toolkits tend to be less sophisticated than viruses designed from scratch, the sheer number of new viruses that can be generated creates a problem for antivirus schemes.

Macro Viruses

In the mid-1990s, macro viruses became by far the most prevalent type of virus. Macro viruses are particularly threatening for a number of reasons:

1. A macro virus is platform independent. Virtually all of the macro viruses infect Microsoft Word documents. Any hardware platform and operating system that supports Word can be infected.
2. Macro viruses infect documents, not executable portions of code. Most of the information introduced onto a computer system is in the form of a document rather than a program.
3. Macro viruses are easily spread. A very common method is by electronic mail.

Macro viruses take advantage of a feature found in Word and other office applications such as Microsoft Excel, namely the macro. In essence, a macro is an executable program embedded in a word processing document or other type of file. Typically, users employ macros to automate repetitive tasks and thereby save keystrokes. The macro language is usually some form of the Basic programming language. A user might define a sequence of keystrokes in a macro and set it up so that the macro is invoked when a function key or special short combination of keys is input.

Successive releases of Word provide increased protection against macro viruses. For example, Microsoft offers an optional Macro Virus Protection tool that detects suspicious Word files and alerts the customer to the potential risk of opening a file with macros. Various antivirus product vendors have also developed tools to detect and correct macro viruses. As in other types of viruses, the arms race continues in the field of macro viruses, but they no longer are the predominant virus threat.

E-mail Viruses

A more recent development in malicious software is the e-mail virus. The first rapidly spreading e-mail viruses, such as Melissa, made use of a Microsoft Word macro embedded in an attachment. If the recipient opens the e-mail attachment, the Word macro is activated. Then

1. The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package.
2. The virus does local damage.

At the end of 1999, a more powerful version of the e-mail virus appeared. This newer version can be activated merely by opening an e-mail that contains the virus rather than opening an attachment. The virus uses the Visual Basic scripting language supported by the e-mail package.

Thus we see a new generation of malware that arrives via e-mail and uses e-mail software features to replicate itself across the Internet. The virus propagates itself as soon as activated (either by opening an e-mail attachment of by opening the e-mail) to all of the e-mail addresses known to the infected host. As a result, whereas viruses used to take months or years to propagate, they now do so in hours. This makes it very difficult for antivirus software to respond before much damage is done. Ultimately, a greater degree of security must be built into Internet utility and application software on PCs to counter the growing threat.

Worms

A worm is a program that can replicate itself and send copies from computer to computer across network connections. Upon arrival, the worm may be activated to replicate and propagate again. In addition to propagation, the worm usually performs some unwanted function. An e-mail virus has some of the characteristics of a worm, because it propagates itself from system to system. However, we can still classify it as a virus because it requires a human to move it forward. A worm actively seeks out more machines to infect and each machine that is infected serves as an automated launching pad for attacks on other machines.

Network worm programs use network connections to spread from system to system. Once active within a system, a network worm can behave as a computer virus or bacteria, or it could implant Trojan horse programs or perform any number of disruptive or destructive actions.

To replicate itself, a network worm uses some sort of network vehicle. Examples include the following:

- Electronic mail facility: A worm mails a copy of itself to other systems.
- Remote execution capability: A worm executes a copy of itself on another system.
- Remote login capability: A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other.

The new copy of the worm program is then run on the remote system where, in addition to any functions that it performs at that system, it continues to spread in the same fashion.

A network worm exhibits the same characteristics as a computer virus: a dormant phase, a propagation phase, a triggering phase, and an execution phase. The propagation phase generally performs the following functions:

1. Search for other systems to infect by examining host tables or similar repositories of remote system addresses.

2. Establish a connection with a remote system.

3. Copy itself to the remote system and cause the copy to be run.

The network worm may also attempt to determine whether a system has previously been infected before copying itself to the system. In a multiprogramming system, it may also disguise its presence by naming itself as a system process or using some other name that may not be noticed by a system operator.

As with viruses, network worms are difficult to counter.

The Morris Worm

Until the current generation of worms, the best known was the worm released onto the Internet by Robert Morris in 1998. The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation. When a copy began execution, its first task was to discover other hosts known to this host that would allow entry from this host. The worm performed this task by examining a variety of lists and tables, including system tables that declared which other machines were trusted by this host, users' mail forwarding files, tables by which users gave themselves permission for access to remote accounts, and from a program that reported the status of network connections. For each discovered host, the worm tried a number of methods for gaining access:

1. It attempted to log on to a remote host as a legitimate user. In this method, the worm first attempted to crack the local password file, and then used the discovered passwords and corresponding user IDs. The assumption was that many users would

use the same password on different systems. To obtain the passwords, the worm ran a password-cracking program that tried
    1. Each user's account name and simple permutations of it
    2. A list of 432 built-in passwords that Morris thought to be likely candidates
    3. All the words in the local system directory
2. It exploited a bug in the finger protocol, which reports the whereabouts of a remote user.
3. It exploited a trapdoor in the debug option of the remote process that receives and sends mail.

If any of these attacks succeeded, the worm achieved communication with the operating system command interpreter. It then sent this interpreter a short bootstrap program, issued a command to execute that program, and then logged off. The bootstrap program then called back the parent program and downloaded the remainder of the worm. The new worm was then executed.

Recent Worm Attacks

The contemporary era of worm threats began with the release of the Code Red worm in July of 2001. Code Red exploits a security hole in the Microsoft Internet Information Server (IIS) to penetrate and spread. It also disables the system file checker in Windows. The worm probes random IP addresses to spread to other hosts. During a certain period of time, it only spreads. It then initiates a denial-of-service attack against a government Web site by flooding the site with packets from numerous hosts. The worm then suspends activities and reactivates periodically. In the second wave of attack, Code Red infected nearly 360,000 servers in 14 hours. In addition to the havoc it causes at the targeted server, Code Red can consume enormous amounts of Internet capacity, disrupting service.

Code Red II is a variant that targets Microsoft IISs. In addition, this newer worm installs a backdoor allowing a hacker to direct activities of victim computers.

In late 2001, a more versatile worm appeared, known as Nimda. Nimda spreads by multiple mechanisms:

- from client to client via e-mail
- from client to client via open network shares
- from Web server to client via browsing of compromised Web sites
- from client to Web server via active scanning for and exploitation of various Microsoft IIS 4.0 / 5.0 directory traversal vulnerabilities
- from client to Web server via scanning for the back doors left behind by the "Code Red II" worms

The worm modifies Web documents (e.g., .htm, .html, and .asp files) and certain executable files found on the systems it infects and creates numerous copies of itself under various filenames.

In early 2003, the SQL Slammer worm appeared. This worm exploited a buffer overflow vulnerability in Microsoft SQL server. The Slammer was extremely compact and spread rapidly, infecting 90% of vulnerable hosts within 10 minutes. Late 2003 saw the arrival of the

Sobig.f worm, which exploited open proxy servers to turn infected machines into spam engines. At its peak, Sobig.f reportedly accounted for one in every 17 messages and produced more than one million copies of itself within the first 24 hours.

Mydoom is a mass-mailing e-mail worm that appeared in 2004. It followed a growing trend of installing a backdoor in infected computers, thereby enabling hackers to gain remote access to data such as passwords and credit card numbers. Mydoom replicated up to 1000 times per minute and reportedly flooded the Internet with 100 million infected messages in 36 hours.

State of Worm Technology

The state of the art in worm technology includes the following:

- **Multiplatform:** Newer worms are not limited to Windows machines but can attack a variety of platforms, especially the popular varieties of UNIX.
- **Multiexploit:** New worms penetrate systems in a variety of ways, using exploits against Web servers, browsers, e-mail, file sharing, and other network-based applications.
- **Ultrafast spreading:** One technique to accelerate the spread of a worm is to conduct a prior Internet scan to accumulate Internet addresses of vulnerable machines.
- **Polymorphic:** To evade detection, skip past filters, and foil real-time analysis, worms adopt the virus polymorphic technique. Each copy of the worm has new code generated on the fly using functionally equivalent instructions and encryption techniques.
- **Metamorphic**: In addition to changing their appearance, metamorphic worms have a repertoire of behavior patterns that are unleashed at different stages of propagation.
- **Transport vehicles:** Because worms can rapidly compromise a large number of systems, they are ideal for spreading other distributed attack tools, such as distributed denial of service zombies.
- **Zero-day exploit:** To achieve maximum surprise and distribution, a worm should exploit an unknown vulnerability that is only discovered by the general network community when the worm is launched

Virus Counter Measure

A countermeasure is an action, process, device, or system that can prevent, or mitigate the effects of, threats to a computer, server or network. In this context, a threat is a potential or actual adverse event that may be malicious or incidental, and that can compromise the assets of an enterprise or the integrity of a computer or network.

Countermeasures cantake the form of software, hardware and modes of behavior. Software countermeasures include:

- personal firewalls

- application firewalls
- anti-virus software
- pop-up blockers
- spyware detection/removal programs.

The most common hardware countermeasure is a router that can prevent the IP address of an individual computer from being directly visible on the Internet. Other hardware countermeasures include:

- biometric authentication systems
- physical restriction of access to computers and peripherals
- intrusion detectors
- alarms.

Behavioral countermeasures include:

- frequent deletion of stored cookies and temporary files from Web browsers
- regular scanning for viruses and other malware
- regular installation of updates and patches for operating systems
- refusing to click on links that appear within e-mail messages
- refraining from opening e-mail messages and attachments from unknown senders
- staying away from questionable Web sites
- regularly backing up data on external media.

In military applications, a countermeasure is a system or strategy intended to prevent an enemy from compromising a target. This can be done by shielding, concealing or moving the target, creating decoys or otherwise confusing the enemy.

FIREWALLS

A firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted internal network and untrusted external network, such as the Internet.

Firewall design principles

Internet connectivity is no longer an option for most organizations. However, while internet access provides benefits to the organization, it enables the outside world to reach and interact with local network assets. This creates the threat to the organization. While it is possible to equip each workstation and server on the premises network with strong security accepted, is the firewall.

The firewall is inserted between the premise network and 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 audit can be imposed. The firewall can be a single computer system or a set of two or more systems that cooperate to perform the firewall function.

Firewall characteristics:

features, such as intrusion protection, this is not a practical approach. The alternative, increasingly

·All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall.

·Various configurations are possible.

·Only authorized traffic, as defined by the local security policy, will be allowed to pass.

·Various types of firewalls are used, which implement various types of security policies.

·The firewall itself is immune to penetration. This implies that use of a trusted system with a secure operating system. This implies that use of a trusted system with a secure operating system.

Four techniques that firewall use to control access and enforce the site‟s security policy is as follows:

1.**Service control –** determines the type of internet services that can be accessed, inbound or outbound. The firewall may filter traffic on this basis of IP address and TCP port number; may provide proxy software that receives and interprets each service request before passing it on; or may host the server software itself, such as web or mail service.

2.**Direction control –** determines the direction in which particular service request may be initiated and allowed to flow through the firewall.

3.**User control –** controls access to a service according to which user is attempting to access it.

4.**Behavior control –** controls how particular services are used.

Capabilities of firewall

A firewall defines a single choke point that keeps unauthorized users out of .

Some of the attacks that can be made on packet filtering routers and the appropriate counter measures are the following:

·**IP address spoofing** – the intruders transmit packets from the outside with a source IP address field containing an address of an internal host.

Countermeasure: to discard packet with an inside source address if the packet arrives on an external interface.

·**Source routing attacks** – the source station specifies the route that a packet should takeas it crosses the internet; i.e., it will bypass the firewall.

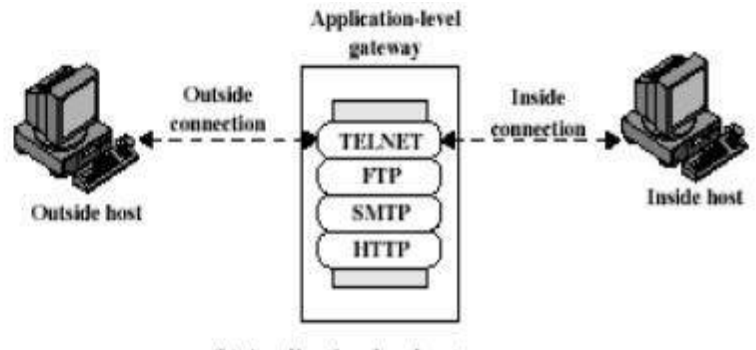
·**Tiny fragment attacks** – the intruder create extremely small fragments and force the TCP header information into a separate packet fragment. The attacker hopes that only the first fragment is examined and the remaining fragments are passed through.

Countermeasure: to discard all packets where the protocol type is TCP and the IP fragment offset is equal to 1.

Application level gateway

An Application level gateway, also called a proxy server, acts as a relay of application level traffic. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints.

Application level gateways tend to be more secure than packet filters. It is easy to log and audit all incoming traffic at the application level. A prime disadvantage is the additional processing overhead on each connection.



(b) Application-level gateway

the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks.

A firewall provides a location for monitoring security related events. Audits and alarms can be implemented on the firewall system.

A firewall is a convenient platform for several internet functions that are not security related.

A firewall can serve as the platform for IPsec.

Limitations of firewall

·The firewall cannot protect against attacks that bypass the firewall. Internal systems may have dial-out capability to connect to an ISP. An internal LAN may support a modem pool that provides dial-in capability for traveling employees and telecommuters.

·The firewall does not protect against internal threats. The firewall does not protect against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.

·The firewall cannot protect against the transfer of virus-infected programs or files. Because of the variety of operating systems and applications supported inside the perimeter, it would be impractical and perhaps impossible for the firewall to scan all incoming files, e-mail, and messages for viruses.

Types of firewalls

There are 3 common types of firewalls.

·Packet filters

·Application-level gateways

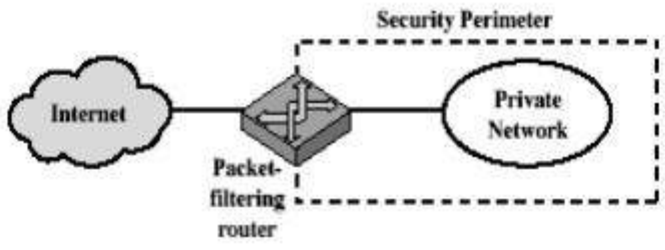
·Circuit-level gateways

Packet filtering router

A packet filtering router applies a set of rules to each incoming IP packet and then forwards or discards the packet. The router is typically configured to filter packets going in both directions. Filtering rules are based on the information contained in a network packet:

·Source IP address – IP address of the system that originated the IP packet.
Destination IP address – IP address of the system, the IP is trying to reach.
Source and destination transport level address – transport level port number.
IP protocol field – defines the transport protocol.

·Interface – for a router with three or more ports, which interface of the router the packet come from or which interface of the router the packet is destined for.



(a) Packet-filtering router

The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken.

Two default policies are possible:

· Default = discard: That which is not expressly permitted is prohibited.

· Default = forward: That which is not expressly prohibited is permitted.

The default discard policy is the more conservative. Initially everything is blocked, and services must be added on a case-by-case basis. This policy is more visible to users, who are most likely to see the firewall as a hindrance. The default forward policy increases ease of use for end users but provides reduced security.

Advantages of packet filter router

- Simple
- Transparent to users
- Very fast

Weakness of packet filter firewalls

·Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application specific vulnerabilities or functions.

.Because of the limited information available to the firewall, the logging functionality present in packet filter firewall is limited.

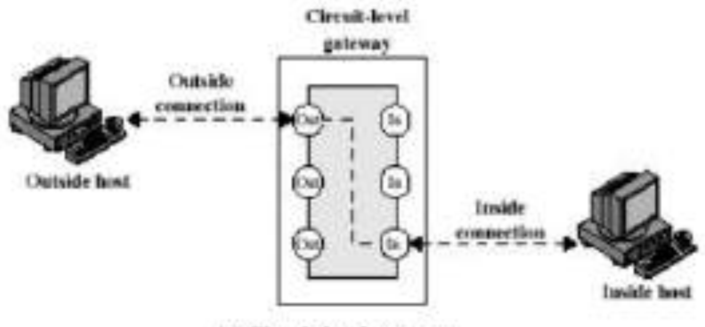
.It does not support advanced user authentication schemes.

·They are generally vulnerable to attacks such as layer address spoofing

Circuit level gateway

Circuit level gateway can be a stand-alone system or it can be a specified function performed by an application level gateway for certain applications. 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 outer 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.



(c) Circuit-level gateway

Basiton host

It is a system identified by the firewall administrator as a critical strong point in the network‟s security. The Bastion host serves as a platform for an application level and circuit level gateway.

Common characteristics of a Basiton host are as follows:

·The Bastion host hardware platform executes a secure version of its operating system, making it a trusted system.

·Only the services that the network administrator considers essential are installed on the Bastion host.

·It may require additional authentication before a user is allowed access to the proxy services.

·Each proxy is configured to support only a subset of standard application‟s command set.

·Each proxy is configured to allow access only to specific host systems.

·Each proxy maintains detailed audit information by logging all traffic, each connection

.and the duration of each connection.

·Each proxy is independent of other proxies on the Bastion host.

·A proxy generally performs no disk access other than to read its initial configuration file.

·Each proxy runs on a non privileged user in a private and secured directory on the Bastion host.

Trusted System:

One way to enhance the ability of a system to defend against intruders and malicious programs is to implement trusted system technology. This section provides a brief overview of this topic. We begin by looking at some basic concepts of data access control.

Data Access Control

Following successful logon, the user has been granted access to one or a set of hosts and applications. This is generally not sufficient for a system that includes sensitive data in its database. Through the user access control procedure, a user can be identified to the system. Associated with each user, there can be a profile that specifies permissible operations and file accesses. The operating system can then enforce rules based on the user profile. The database management system, however, must control access to specific records or even portions of records. For example, it may be permissible for anyone in administration to obtain a list of company personnel, but only selected individuals may have access to salary information. The issue is more than just one of level of detail. Whereas the operating system may grant a user permission to access a file or use an application, following which there are no further security checks, the database management system must make a decision on each individual access attempt. That decision will depend not only on the user's identity but also on the specific parts of the data being accessed and even on the information already divulged to the user.

A general model of access control as exercised by a file or database management system is that of an **access matrix** (Figure 20.3a). The basic elements of the model are as follows:

- **Subject**: An entity capable of accessing objects. Generally, the concept of subject equates with that of process. Any user or application actually gains access to an object by means of a process that represents that user or application.
- **Object:** Anything to which access is controlled. Examples include files, portions of files, programs, and segments of memory.
- **Access right**: The way in which an object is accessed by a subject. Examples are read, write, and execute.

The Concept of Trusted Systems

Much of what we have discussed so far has been concerned with protecting a given message or item from passive or active attacks by a given user. A somewhat different but widely applicable requirement is to protect data or resources on the basis of levels of security. This is commonly found in the military,

where information is categorized as

- unclassified (U),
- confidential (C),
- secret (S),
- top secret (TS), or beyond.

This concept is equally applicable in other areas, where information can be organized into gross categories and users can be granted clearances to access certain categories of data. For example, the highest level of security might be for strategic corporate planning documents and data, accessible by only corporate officers and their staff; next might come sensitive financial and personnel data, accessible only by administration personnel, corporate officers, and so on.

When multiple categories or levels of data are defined, the requirement is referred to as **multilevel security**. The general statement of the requirement for multilevel security is that a subject at a high level may not convey information to a subject at a lower or noncomparable level unless that flow accurately reflects the will of an authorized user. For implementation purposes, this requirement is in two parts and is simply stated. A multilevel secure system must enforce the following:

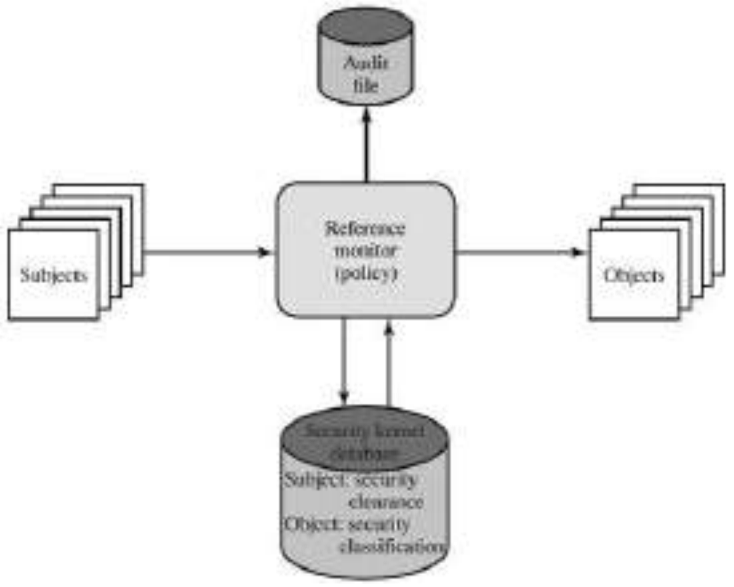
- No read up: A subject can only read an object of less or equal security level. This is referred to in the literature as the Simple Security Property.
- No write down: A subject can only write into an object of greater or equal security level. This is referred to in the literature as the *-Property[1](pronounced star property).

  [1] The "*" does not stand for anything. No one could think of an appropriate name for the property during the writing of the first report on the model. The asterisk was a dummy character entered in the draft so that a text editor could rapidly find and replace all instances of its use once the property was named. No name was ever devised, and so the report was published with the "*" intact.

These two rules, if properly enforced, provide multilevel security. For a data processing system, the approach that has been taken, and has been the object of much research and development, is based on the *reference monitor* concept. This approach is depicted in Figure 20.4. The reference monitor is a controlling element in the hardware and operating system of a computer that regulates the access of subjects to objects on the basis of security parameters of the subject and object. The reference monitor has access to a file, known as the security kernel database, that lists the access privileges (security clearance) of each subject and the protection attributes (classification level) of each object. The reference monitor enforces the security rules (no read up, no write down) and has the following properties:

- Complete mediation: The security rules are enforced on every access, not just, for example, when a file is opened.
- Isolation: The reference monitor and database are protected from unauthorized modification.
- Verifiability: The reference monitor's correctness must be provable. That is, it must be possible to demonstrate mathematically that the reference monitor enforces the security rules and provides complete mediation and isolation.

Reference Monitor Concept


Audit file
Subjects
Reference monitor (policy)
Objects
Security kernel database
Subject: security clearance
Object: security classification

These are stiff requirements. The requirement for complete mediation means that every access to data within main memory and on disk and tape must be mediated. Pure software implementations impose too high a performance penalty to be practical; the solution must be at least partly in hardware. The requirement for isolation means that it must not be possible for an attacker, no matter how clever, to change the logic of the reference monitor or the contents of the security kernel database. Finally, the requirement for mathematical proof is formidable for something as complex as a general-purpose computer. A system that can provide such verification is referred to as a **trusted system**.

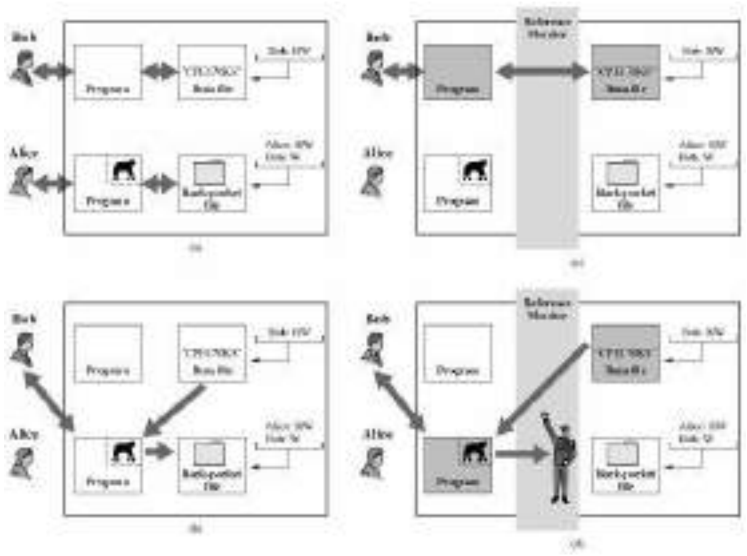
A final element illustrated in Figure 20.4 is an audit file. Important security events, such as detected security violations and authorized changes to the security kernel database, are stored in the audit file.

In an effort to meet its own needs and as a service to the public, the U.S. Department of Defense in 1981 established the Computer Security Center within the National Security Agency (NSA) with the goal of encouraging the widespread availability of trusted computer systems. This goal is realized through the center's Commercial Product Evaluation Program. In essence, the center attempts to evaluate commercially available products as meeting the security requirements just outlined. The center classifies evaluated products according to the range of security features that they provide. These evaluations are needed for Department of Defense procurements but are published and freely available. Hence, they can serve as guidance to commercial customers for the purchase of commercially available, off-the-shelf equipment.

Trojan Horse Defense

One way to secure against Trojan horse attacks is the use of a secure, trusted operating system. Figure 20.5 illustrates an example. In this case, a Trojan horse is used to get around the standard security mechanism used by most file management and operating systems: the access control list. In this example, a user named Bob interacts through a program with a data file containing the critically sensitive character string "CPE170KS." User Bob has created the file with read/write permission provided only to programs executing on his own behalf: that is, only processes that are owned by Bob may access the file.

Trojan Horse and Secure Operating System



The Trojan horse attack begins when a hostile user, named Alice, gains legitimate access to the system and installs both a Trojan horse program and a private file to be used in the attack as a "back pocket." Alice gives read/write permission to herself for this file and gives Bob write-only permission (Figure 20.5a). Alice now induces Bob to invoke the Trojan horse program, perhaps by advertising it as a useful utility. When the program detects that it is being executed by Bob, it reads the sensitive character string from Bob's file and copies it into Alice's back-pocket file (Figure 20.5b). Both the read and write operations satisfy the constraints imposed by access control lists. Alice then has only to access Bob's file at a later time to learn the value of the string.

Now consider the use of a secure operating system in this scenario. Security levels are assigned to subjects at logon on the basis of criteria such as the terminal from which the computer is being accessed and the user involved, as identified by password/ID. In this example, there are two security levels, sensitive and public, ordered so that sensitive is higher than public. Processes owned by Bob and Bob's data file are assigned the security level sensitive. Alice's file and processes are restricted to public. If Bob invokes the Trojan horse program , that program acquires Bob's security level. It is therefore able, under the simple security property, to observe the sensitive character string. When the program attempts to store the string in a public file (the back-pocket file), however, the is violated and the attempt is disallowed by the reference monitor. Thus, the attempt to write into the back-pocket file is denied even though the access control list permits it: The security policy takes precedence over the access control list mechanism.