5. Application Layer

① DNS — The Domain Name System:-

→ Programs theoretically could refer to hosts, mailbo & other resources by their network (eg:- Ip) addresses, the addresses are hard for People to remember.

Eg:- tana @ 128.111.24.41
↓ ↓
ISP E-mail address

→ So, Ascii names were introduced to decoupl machine names from machine addresses.

Eg:- tana @ art.uscb.Edu

→ When thousands of minicompulers & pc's were Connected to the net, Everyone realized that this approach Could not Continue forever, bcz the file size would bec too large, host name Conflicts etc. To Solve this DNS (Domain Name System) was inverted.

→ DNS is a hierarichal, domain-based nami Scheme & a distributed database System for implementi this Scheme.

→ DNS is Primarily used for mapping hosts names & E-mail destinations to Ip addresses but can also used for other Purposes

→ DNS is defined in RFC's 1034 and Rfc's 1035.

→ To map a name onto an Ip address, an applicatio Program calls a library Procedure called the resolver, Passing it the name as a Parameter. (i.e get host by name),

→ Now, the resolver sends a UDP Packet to local DNS Server, which then looks up the name & returns the Ip address to the resolver, which then returns it to the caller. The Program then Establish Tcp Connection with the destination or Send it UDP Packets.

DNS Name Space:-

→ The Internet is divided into Over 200 top-level domains, where Each domain Covers many hosts. Each domain is Partitioned into Subdomains & further so on.

→ The top-level domains Comes in two flavors: generic & Countries.

→ The Original generic domains wer Com (Commercial) Edu (Educational institutions), gov (the U.S federal government), int (certain international Organizations), net (network Providers), & org (non-profit) and Org (nonprofit Organizations).

→ Each domain is named by Path upward from it to the (Unnamed) root. The Components are Seperated by Periods (Pronounced "dot")

Eg:-
Eng.Sun.Com, jack.ac.m.org etc.

→ Domain names can be either absolute or relative.

→ An absolute domain name always ends with a Period (Eg., Eng.Sun.Com), where as relative one does not.

→ Domain names are case insensitive

Eg:- Edu, Edu or EDU → same meaning.

(5) Resource Records:-

→ Every domain, whether it is a single host or a top-level domain, can have a set of resource record associated with it.

→ Resource record is a five tuple.

Domain_name	Time_to_live	class	Type	value

- Domain name:-
  → It tells the domain to which this record appli

- Time-to-live:-
  → It gives an ~~idle~~ indication of how stable the record is.

Eg:- . highly stable → 86400 → 1 day.
. Small value → 60 → 1 minute.

- class:-
  → for internet information it is always IN.
  → for non-Internet information, other codes can be used, but rarely they are used.

Type:-

→ It tells what kind of record this is.

Eg:-

Type	Meaning	Value.
SOA	Start of Authority	Parameter for this Zone
A	Ip address of a host	32-bit integer
MX	Mail Exchange	Priority, domain willing to accept E-mail
NS	Name Server	Name of a Server for this Domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an Ipaddress
HINFO	Host description	CPU & OS in ASCII
Txt	Text	Uninterpreted ASCII text.

Name Servers:-

→ To avoid Problems associated with having only a Single Source of information, the DNS name Space is divided into nonoverlapping Zones. One Possible way is shown in fig ② i.e we divide the name Space.

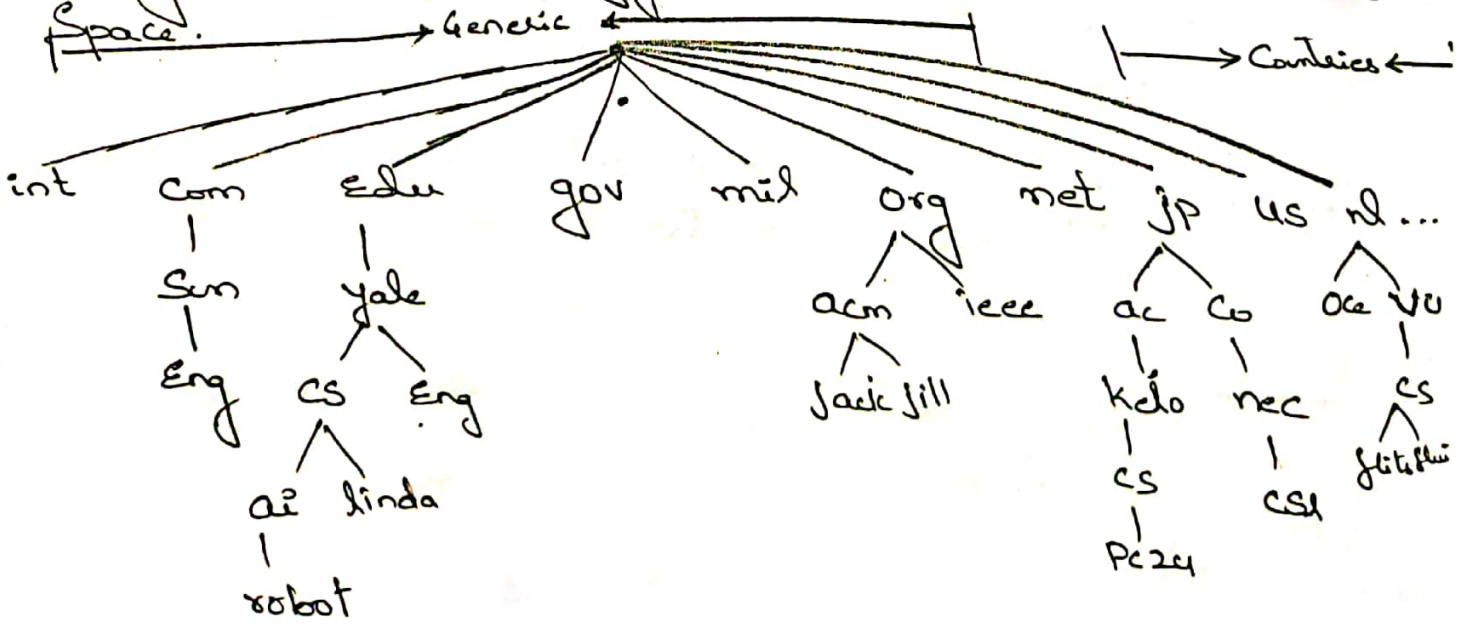


fig ②:- A Portion of the Internet domain name Space.

→ We divide namespaces in fig (6).

fig (6) Part of the DNS name Spacing Showing the division into Zones.

→ Each Zone Contains Some Part of the tree & also Contains name Servers holding the information about that Zone.

→ Normally, a Zone will have One Primary name Server, which gets its information from a file on its di & One or more Secondary name Servers, which gets their information from their Primary name Server.

→ When a resolver has a query about a doma name, it Passes the query to One of the local name Servers, such as ai.cs.yale.Edu falling under cs.yale it returns the authoritative resource records.

→ An authoritative record is One that comes from the authority that manages the record. & is thus always Correct.

→ If the domain is ~~route~~ remote & has no information about the requested domain is available locally, the

name Server sends a query message to the top-level name Server for domain requested!

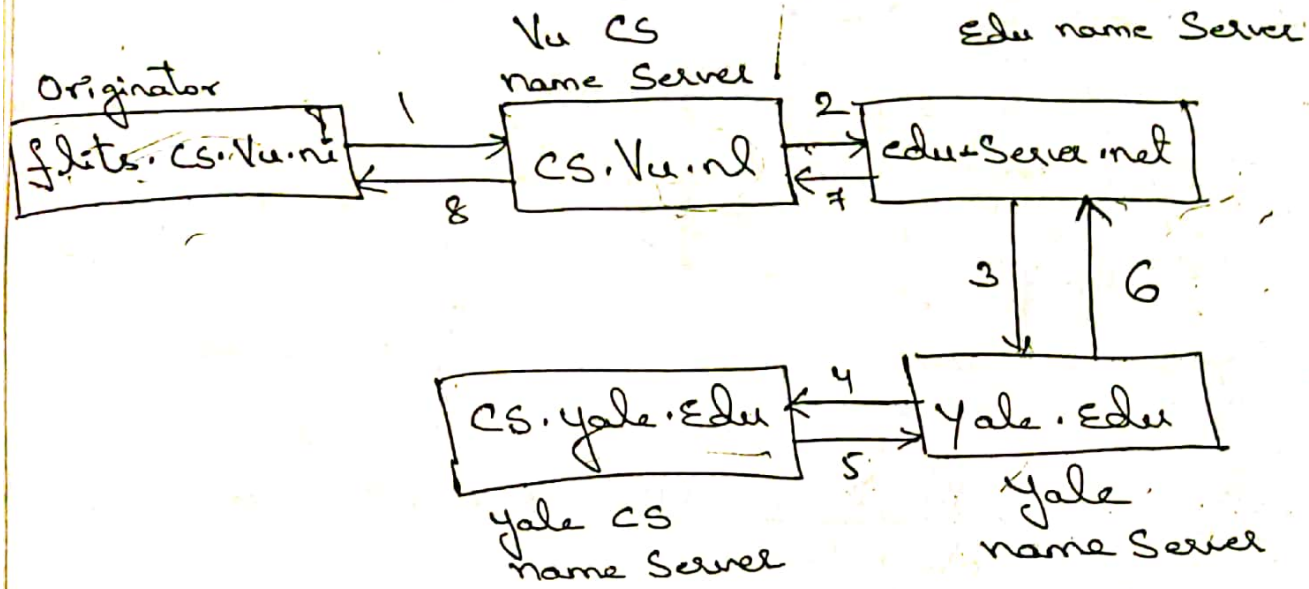


fig (c) :- How a resolver looks up a remote name in Eight steps.

2. Electronic mail:—

→ In olden days, if we wants transfer messages is done through Paper (i.e letters, faxes etc). Since it is a very slow.

→ In 1990's to reduce time, money, & to send message in Efficient way E-mails (Electronic mails) came into Existance.

→ E-mails are is full of jargon such as BTW (By the Way), ROTFL (Rolling On the Floor Laughing) & IMHO (In My Humble Opinion).

→ Many People also use little ASCII Symbols called Smiley's or Emoticons in their E-mails.

→ The first E-mail System simply Consisted of file transfer Protocols.

→ Some of the Complaints are:-

1. Sending a message to a group of People was inConveni Managers often needs this facility to send memos to a their Subordinates.
2. The Originator (Sends) never knew if a message arrived or not.
3. If SomeOne was planning to be away on business for Several weeks & wanted all incoming E-mail to be handled by his Secretary, this was not so Easy to arrange.
4. It is not Possible to Create & Send messages Contain a mixture of text, drawings, facsimile & Voice.

→ In 1982, the ARPANET E-mail Proposals were Published as RFC 821 (transmission Protocol) and RFC 822 (message format).

→ Minor revisions, RFC 2821 & RFC 2822 have become internet Standards, but Everyone still refers to Internet E-mail as RFC 822.

→ In 1984, CCITT drafted its X.400 recommendation. After 2-decades of Competitions, E-mail System based on RFC 822 are widely used, whereas those based on X.400 have disappread.

② Architecture & Services:-

→ E-mail architecture Consists of two subsystems. They are:

1. User agents
2. Message transfer agents.

User agents:-

→ It allows People to read & Send E-mails.

→ User agents are local People. Programs that Provide a Command-based, menu-based or graphical method for interacting with the E-mail System.

Message transfer agents:-

→ It moves the messages from Source to the destinations.

→ These agents are typically System daemons i.e Processes that run in the background. Their job is to move E-mail through the systems.

→ E-mail system supports five basic functions. They are:-

(a) Composition — refers to the Process of Creating messages & answers

(b) Transfer — moving message from Originator to the Recipient.

(c) Reporting — has to do with telling the Originator what happend to the message

Delivered, rejected, lost.

- Displaying — incoming message is needed so people can read their E-mail.

- Disposition — It is the final step & Concerns what the receipt does with the message after receiving it.

→ Most systems allows users to create mailb to store incoming E-mail. Commands are needed to create & destroy mailboxes, inspect the contents of mailboxes, insert & delete messages from mailboxes & so on.

→ A key idea in E-mail system is the distinc between the Envelope & its contents.

→ Envelope Encapsulates the message. It contains all the information needed for transporting the message such as destination address, Priority & Securit level, ~~all of~~

→ The message inside the Envelope Consists of two parts

- header → Control information of user agen
- body → Entirely for human recipient

User Agents :-

→ A User agent is normally a Program (Sometime called a mail reader) that accepts a Variety of Command. for Composing, receiving & replying to messages as well as for manipulating mailboxes.

→ Some User agents have a fancy menu or icons driven interface that requires a mouse, keyboard, command Etc.

→ Some Systems are menu or icon-driven but also have keyboard shortcuts.

Sending E-mail:

→ To Send an E-mail message, a User must Provide the message, the destination address & Possibly Some other Parameters.

→ Messages can be Produced with text-Editor, MS-word, Word-Processing Program etc.

→ The destination address must be in form of the DNS address, such as:

User @dns-address.

→ The other format of address also Exit such as X.400 which is different from DNS addresses. They are Composed of attribute = Value Pairs Seperated by Slashes.

Eg:-

/C = US (C → Country) / ST = MASSACHUSETTS (ST → state) / L = CAMBRIDGE (L → locality) / PA = 360 MEMORIAL DR. (PA → Personal address) / CN = KEN SMITH (CN → Common name) /.

Where:-

→ Most E-mail Support mailing lists, so that a User can send the same message to a list of People with a Single Command.

→ If the mailing list is maintained locally, the User agent can just send a Separate message to Each intended recipient.

* Reading E-mail:-

→ The User's mailbox for incoming E-mail before displaying anything on the Screen.

→ Then it may announce the no. of messages in the mailbox or display a One-line Summary of Each one & wait for a Command.

→ Mailbox Contains eight messages, such as

#	Flags	Bytes	Sender	Subject
1.	K	1030	ASW	changes to MINIX
2	KA	6348	trudy	Not all Trudys are nasty
3	KF	4519	Amy N. Wong	Request for information
4.		1236	bal	Bio informatics
5		104110	Kaashoek	Material on Peer-to-Peer
6		1223	Frank	Re: Will you review a grant Proposal
7		3110	guido	our Paper has been accepted
8		1204	dmr	Re: My Student's visit

5. Message formats :-

→ E-mail message formats are RFC 822 & multimedia extensions to RFC 822.

RFC 822 :-

→ Messages Consist of a Primitive Envelope, Some no. of header fields, a blank line & then message body.

→ RFC 822 was designed decades ago & does not clearly distinguish the Envelope fields from the header fields.

→ Although it was revised in RFC 2822, Completely redoing it was not Possible due to its widespread Usage.

→ The Principle header fields related to message transport are:

To :- field gives the DNS address of Primary recipient

CC :- field gives the address of Secondary receiptient

BCC (Blind Carbon Copy) :- field is like CC, Except that this line is deleted from all copies sent to the Primary & Secondary recipients.

From & Sender :- tells who wrote & Sent the message respectively.

Received :- It is added by Each transfer agent along the way. The line Contains agent's identity, date & time the message was received, & other information that can be used for finding bugs in the routing System.

Return-Path :- This field is added by final message transfer agent & was intended to tell how to get back to the Sender.

Header	Meaning
To:	E-mail addresses of Primary recipient
Cc:	E-mail addresses of Secondary "
Bcc:	" " for blind Carbon copy
From:	Person or People who Created the message
Sender:	E-mail address of the actual sender
Received:	Line added by Each transfer agent along the route
Return-Path:	Can be Used to identify a path back to the Sender.

fig: RFC 822 header fields related to message transport

→ In addition to above fields, RFC 822 mess contains may also Contain a variety of header fields Used by user agents or human recipients. They a

* Reply-To: field is Sometimes used when neither the Per Composing the message nor the Person Sendi the message wants to see the reply

fig: Some fields used in the RFC 822 message header

Header	Meaning
Date:	The date & time the message was sent
Reply-To:	E-mail address to which replies should be sent
Message-Id:	Unique number for referencing this message later
In-Reply To:	Message-Id of the message to which this is a reply
References:	other relevant message-Id's
Keywords:	User-chosen keywords
Subject:	Short Summary of the message for the one-line deliv

MIME (Multipurpose Internet Mail Extensions):-

→ In earlier/olden days of ARPANET, E-mail was exclusively used for text messages written in English Expressed in ASCII. So, in that we used RFC 822 to format. The Problem include Sending & receiving is:-

1. Messages in languages with accents (Eg:- French & German)
2. Messages in non-Latin alphabets (Eg:- Hebrew & Russian)
3. Messages in languages without alphabets (Eg:- Chinese & Japanese)
4. Messages not Containing text at all (Eg:- audio or images)

→ A solution was Proposed in RFC 1341 & updated in RFC's 2045 - 2049. This solution is called MIME.

→ The basic idea of MIME is to continue to use the RFC 822 format, but to add Structure to the message body & define Encoding rules for non-ASCII messages.

→ MIME defines five new message headers, They are:-

MIME Version — It is assumed to be an English plaintext message & Processed

* Content - Description — It is an ASCII string telling what is the message.

* Content - Id :- It identifies the Content. It uses the same format as the Message-Id

* Content - Transfer - Encoding :- tells how body is wrapped for transmission through a network that may object to most characters other than letters, numbers & punctuation.

* Content-Type : ~~Spe~~ tells/specifies the type of the file

Eg:- ① Text < Plain, Enriched ; ② Image < GIF, Jpeg

③ Audio - basic ④ Video - Mpeg

④ Application < Octet-Stream, PostScript, ~~RTF~~ ⑤ Message < RFC 822, Partio, Extern body

⑥ Multipart < Alternative, Parallel, Digest

Header	Meaning
MIME version:	Identifies the MIME version.
Content-Description:	Human readable string telling what is in the message
Content-Id:	Unique Identifier
Content-Transfer-Encoding:	How the body is wrapped for the transmission
Content-Type:	Type & format of the content

fig:- RFC 822 header added by MIME

4. Message Transfer:-

→ Message Transfer System is concerned with relaying messages from the Originator to the recipient.

* SMTP:- (Simple Mail Transfer Protocol):-

→ Within the Internet, E-mail is delivered by having the source machine establish TCP connection to Port 25 of the destination machine.

→ Listening to this Port is an E-mail daemon that Speaks SMTP. This daemon ~~Speaks~~ accepts incoming Connection & Copies messages from them into a appropriate mailboxes.

→ SMTP is Simple ASCII Protocol.

→ After Establishing the TCP Connection to Port 25 the Sending machine, Operating as client, waits for the receiving machine, Operating as Server, to talk first.

→ The Server starts by Sending a line of text giving its identity & telling whether it is Prepared to receive mail. If it is not, the client releases the Connection & tries again later.

Final Delivery:-

→ Till now, if we want to send a message to Communicate the Connection (i.e TCP/UDP) is Established between Sender & Receiver.

→ But, People who access the Internet by Calling their ISP Over a modem, it breaks-down. The Problem ——— what be (S) (R) → ? online or not
↓ Connection Established or not.

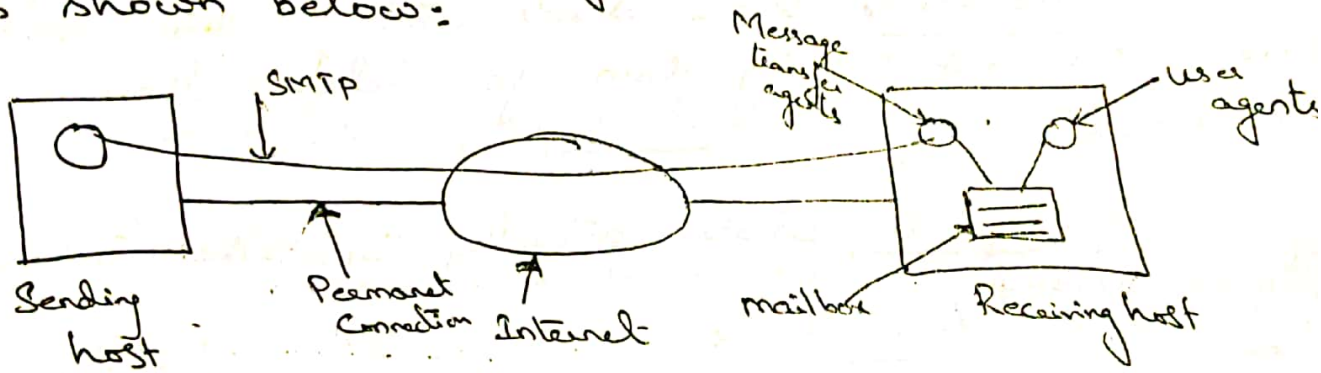
→ One solution is to have a message transfer agents on an ISP machine accept E-mail for its Customers & store it in their mailboxes on an ISP machine. Since this agent can be on-line all the time, E-mail can be sent to it 24 hours a day.

* Pop3 :-

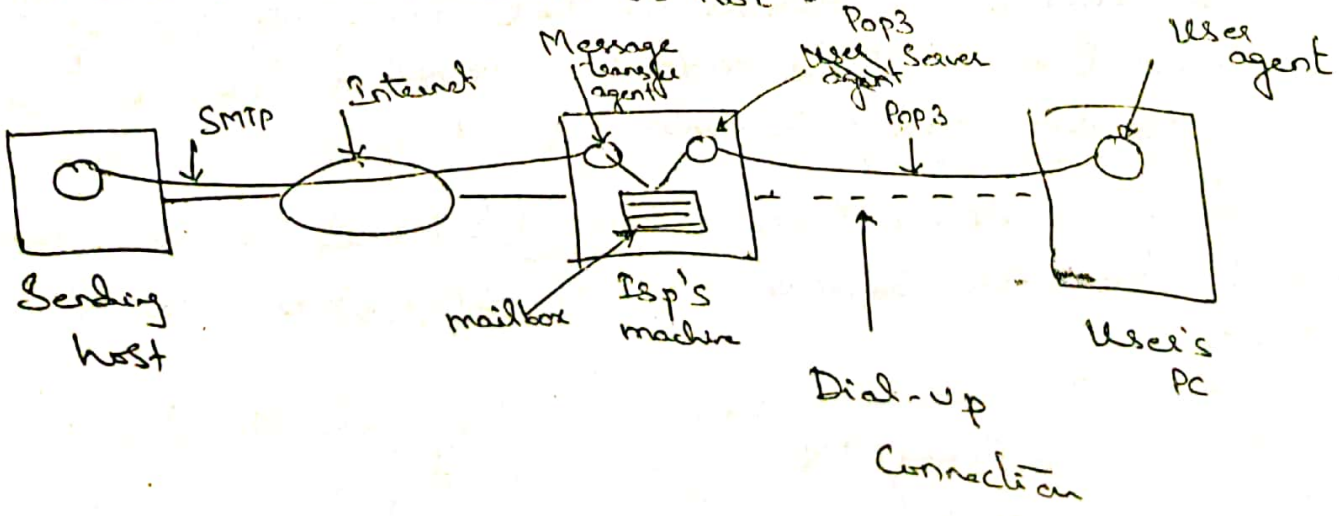
→ Unfortunately, this solution creates another Problem: how does the User get the E-mail from the ISP's messa transfer agents? (on another

→ Solution to this Problem is to Create a Proto that allows user transfer agents (on client Pc's) to Contact the message transfer agent (on Isp's machi & allow E-mail to be Copied from the Isp to the User. One such Protocol is Pop3 (Post-Office Protocol version 3), which is described in RFC 193

→ The Situation that used to hold (both Sender & receiver having Permanent Internet Connecti as shown below:



→ The below fig (b), here Sender is (current on-line but the receiver is not.



→ Pop3 begins when the user starts the mail reader.

→ The mail reader calls up the Isp (unless there is already a Connection) & Establishes a Tcp Connection with message transfer agent as Port 110.

→ Once the Connection has been Established, the Pop3 Protocol goes through three states in Sequence:

1. Authorization
2. Transactions
3. Update.

→ The authorization State deals with having the user log in.

→ Transaction State deals with the user Collecting the E-mails & making them for deletion from the mail box.

→ Update State actually Causes the E-mails to be deleted.

IMAP:-

→ Pop3, normally downloads all stored messages at each Contact, the result is that user's E-mail quickly gets Spread Over multiple machine, more or less random, Some of the them are not Even the Users.

→ To Overcome this IMAP (Internet Message Access Protocol), which is defined in RFC 2060, has Come into Existance.

→ Pop3, basically assumes that user will clearly out the mail box on Every Contact & work offline after that, IMAP assumes that all E-mail will return on the Server indefinitely in multiple mailboxes.

→ IMAP Provides mechanism for Creating, destroying E-

Simple Network Management Protocol (SNMP):-

→ If an Organization has 1000 of devices, then to check all devices, One by One Everyday, are working Properly or not is a hetic task.

→ To Ease these up, Simple Network Management Protocol (SNMP) is Used.

→ SNMP is an application layer Protocol which Uses UDP Port number 161/162.

→ SNMP is used to monitor network, detect network faults and Sometimes Even Used to Configure remote devices.

→ SNMP, Components are of 3-types. They are

1. SNMP Manager:-
   → It is a Centralised System used to monitor network. It is also known as Network Management Station (NMS).

2. SNMP agent:-
   → It is a Software Management i.e Software module installed on a managed device.
   → Managed devices Can be network devices likes pc, router, Switches, Servers etc.

3. Management Information Base:-
   → MIB Consists of information of resources that are to be managed.

→ These information is Organised hierarchically.

→ It Consists of objects instances which are Essential Variables.

→ SNMP messages Consists of different variables such as:

1. Get Request:-

→ SNMP manager sends this message to request data from SNMP agent.

→ It is simply used to retrieve data from SNMP agent.

→ In response to this, SNMP agent responds with requested value through response message.

2. Get Next Request:-

→ This message can be sent to discover what data is available on SNMP agent.

→ SNMP manager can request for data continously until no more data is left.

→ So, that SNMP manager can take knowledge of all available data on SNMP agent.

3. Get Bulk Request:-

→ This message is used to retrieve the large data at Once by the SNMP manager from SNMP agent. It is introduced in SNMP V2C.

4. Set Request:-

→ It is used by SNMP manager to set the value of an object instance on the SNMP agent.

5. Response:-

→ It is a message send from agent upon a request from manager.

→ When sent in response to Get messages, it will contain the data requested.

→ When sent in response to Set message, it will contain the newly set values as confirmation that the value has been set.

6. Trap:-

→ These are the message send by the agent without being requested by the manager.

→ It is sent when a fault has occured.

7. Inform Request:-

→ It is introduced in SNMPV2c, used to identify if the trap message has been received by the manager or not.

→ The agents can be configured to set trap continously until it receives an inform message.

→ It is same as trap but adds an acknowledgement that trap doesn't provide.

* SNMP Security levels:-

→ It defines the type of security algorithm performed on SNMP packets. These are used only on SNMP V3.

→ There are 3 security levels namely:-

1. noAuthNoPriv:-

→ This (no authentication, no Privacy) Security level uses Community String for authentication & no Encyption for Privacy.

2. authNoPriv:-

→ This Security level (authentication, no Privacy) uses HMAC with MD5 for authentication & no Encryptio is used for Privacy.

3. authPriv:-

→ This Security level (authentication, Privacy) uses HMAC with MD5 or SHA for authentication & Encryptia uses DES-56 algorithm.

* SNMP Versions:-

→ There are 3 Versions of SNMP. They are:

1. SNMP V1:

→ It uses Community Strings for authentication & Use UDP only.

2. SNMP V2C:

→ It uses Community Strings for authentication-It Uses UDP but can be Configured to use Tcp.

3. SNMP V3:-

→ It uses Hash based MAC with MD5 or SHA for authentication & DES-56 for Privacy.

→ This Version Uses Tcp.

→ This is more Secure, Compared to above. ~~other~~ Versions.

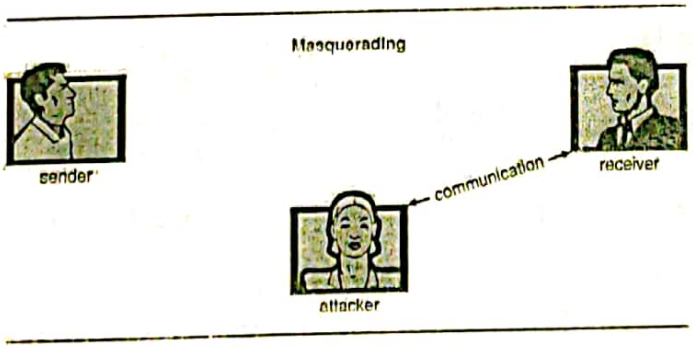
The Security Problem

Protection dealt with protecting files and other resources from accidental misuse by cooperating users sharing a system, generally using the computer for normal purposes. Security deals with protecting systems from deliberate attacks, either internal or external, from individuals intentionally attempting to steal information, damage information, or otherwise deliberately wreak havoc in some manner.

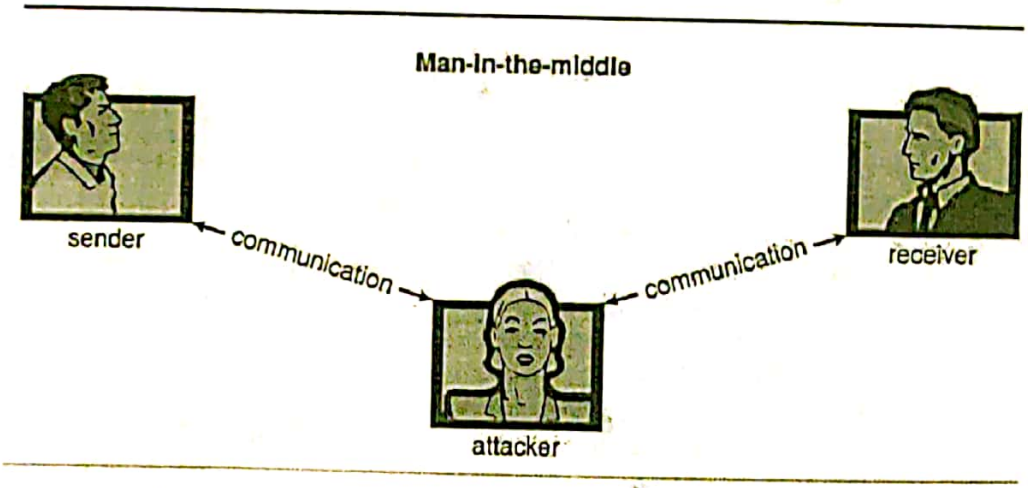
Some of the most common types of ***violations*** include:

- ***Breach of Confidentiality*** - Theft of private or confidential information, such as credit-card numbers, trade secrets, patents, secret formulas, manufacturing procedures, medical information, financial information, etc.
- ***Breach of Integrity*** - Unauthorized ***modification*** of data, which may have serious indirect consequences. For example a popular game or other program's source code could be modified to open up security holes on users systems before being released to the public.
- ***Breach of Availability*** - Unauthorized ***destruction*** of data, often just for the "fun" of causing havoc and for bragging rites. Vandalism of web sites is a common form of this violation.
- ***Theft of Service*** - Unauthorized use of resources, such as theft of CPU cycles, installation of daemons running an unauthorized file server, or tapping into the target's telephone or networking services.
- ***Denial of Service, DOS*** - Preventing legitimate users from using the system, often by overloading and overwhelming the system with an excess of requests for service.

One common attack is ***masquerading,*** in which the attacker pretends to be a trusted third party. A variation of this is the ***man-in-the-middle,*** in which the attacker masquerades as both ends of the conversation to two targets.



201/214



A ***replay attack*** involves repeating a valid transmission. Sometimes this can be the entire attack, (such as repeating a request for a money transfer), or other times the content of the original message is replaced with malicious content.

There are four levels at which a system must be protected:

Physical.- The easiest way to steal data is to pocket the backup tapes. Also, access to the root console will often give the user special privileges, such as rebooting the system as root from removable media. Even general access to terminals in a computer room offers some opportunities for an attacker, although today's modern high-speed networking environment provides more and more opportunities for remote attacks.

Human - There is some concern that the humans who are allowed access to a system be trustworthy, and that they cannot be coerced into breaching security. However more and more attacks today are made via ***social engineering,*** which basically means fooling trustworthy people into accidentally breaching security.

- **Phishing** involves sending an innocent-looking e-mail or web site designed to fool people into revealing confidential information. E.g. spam e-mails pretending to be from e-Bay, PayPal, or any of a number of banks or credit-card companies.

Dumpster Diving involves searching the trash or other locations for passwords that are written down.

Password Cracking involves divining users passwords, either by watching them type in their passwords, knowing something about them like their pet's names, or simply trying all words in common dictionaries.

Operating System - The OS must protect itself from security breaches, such as runaway processes (denial of service), memory-access violations, stack overflow violations, the launching of programs with excessive privileges, and many others.

Network - As network communications become ever more important and pervasive in modern computing environments, it becomes ever more important to protect this area of the system

202/214

➢ Program Threats

There are many common threats to moderr systems. Only a few are discussed here.

- Trojan horse
- Trap Door
- Logic Bomb
- Stack and Buffer Overflow
- Viruses

1. *Trojan Horse*

A ***Trojan Horse*** is a program that secretly performs some maliciousness in addition to its visible actions.
Some Trojan horses are deliberately written as such, and others are the result of legitimate programs that have become infected with *viruses*, (see below.)
One dangerous opening for Trojan horses is long search paths, and in particular paths which include the current directory (".") as part of the path. If a dangerous program having the same name as a legitimate program (or a common mis-spelling, such as "sl" instead of "ls") is placed anywhere on the path, then an unsuspecting user may be fooled into running the wrong program by mistake.
Spyware is a version of a Trojan Horse that is often included in "free" software downloaded off the Internet. Spyware programs generate pop-up browser windows, and may also accumulate information about the user and deliver it to some central site. (This is an example of ***covert channels,*** in which surreptitious communications occur.) Another common task of spyware is to send out spam e-mail messages, which then purportedly come from the infected user.

2. *Trap Door*

A ***Trap Door*** is when a designer or a programmer (or hacker) deliberately inserts a security hole that they can use later to access the system.
Because of the possibility of trap doors, once a system has been in an untrustworthy state, that system can never be trusted again. Even the backup tapes may contain a copy of some cleverly hidden back door.
A clever trap door could be inserted into a compiler, so that any programs compiled with that compiler would contain a security hole. This is especially dangerous, because inspection of the code being compiled would not reveal any problems.

3. *Logic Bomb*

A ***Logic Bomb*** is code that is not designed to cause havoc all the time, but only when a certain set of circumstances occurs, such as when a particular date or time is reached or some other noticeable event.

A classic example is the ***Dead-Man Switch***, which is designed to check whether a certain person (e.g. the author) is logging in every day, and if they don't log in for a long time (presumably because they've been fired), then the logic bomb goes off and either opens up security holes or causes other problems.

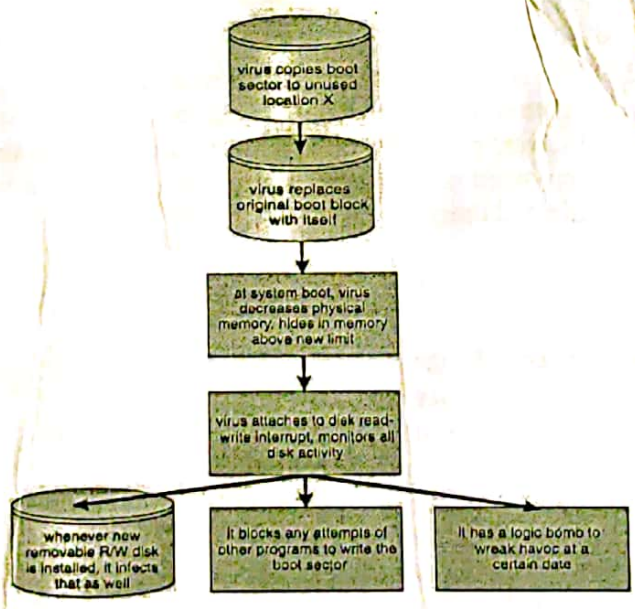
4. *Stack and Buffer Overflow*

This is a classic method of attack, which exploits bugs in system code that allows buffers to overflow.

5. *Viruses*

A virus is a fragment of code embedded in an otherwise legitimate program, designed to replicate itself (by infecting other programs), and (eventually) wreaking havoc.
Viruses are more likely to infect PCs than UNIX or other multi-user systems, because programs in the latter systems have limited authority to modify other programs or to access critical system structures (such as the boot block)
Viruses are delivered to systems in a ***virus dropper,*** usually some form of a Trojan Horse, and usually via e-mail or unsafe downloads.

Figure shows typical operation of a boot sector virus:



Some of the forms of viruses include:

File - A file virus attaches itself to an executable file, causing it to run the virus code first and then jump to the start of the original program. These viruses are termed ***parasitic,*** because they do not leave any new files on the system, and the original program is still fully functional.

Boot - A boot virus occupies the boot sector, and runs before the OS is loaded. These are also known as ***memory viruses***, because in operation they reside in memory, and do not appear in the file system.

Macro - These viruses exist as a macro (script) that are run automatically by certain macro-capable programs such as MS Word or Excel. These viruses can exist in word processing documents or spreadsheet files.

Source code viruses look for source code and infect it in order to spread.

Polymorphic viruses change every time they spread - Not their underlying functionality, but just their ***signature,*** by which virus checkers recognize them.

Encrypted viruses travel in encrypted form to escape detection. In practice they are self-decrypting, which then allows them to infect other files.

Stealth viruses try to avoid detection by modifying parts of the system that could be used to detect it. For example the read( ) system call could be modified so that if an infected file is read the infected part gets skipped and the reader would see the original unadulterated file.

Tunneling viruses attempt to avoid detection by inserting themselves into the interrupt handler chain, or into device drivers.

Multipartite viruses attack multiple parts of the system, such as files, boot sector, and memory.

➢ System and Network Threats

Most of the threats described above are termed ***program threats***, because they attack specific programs or are carried and distributed in programs. The threats in this section attack the operating system or the network itself, or leverage those systems to launch their attacks.

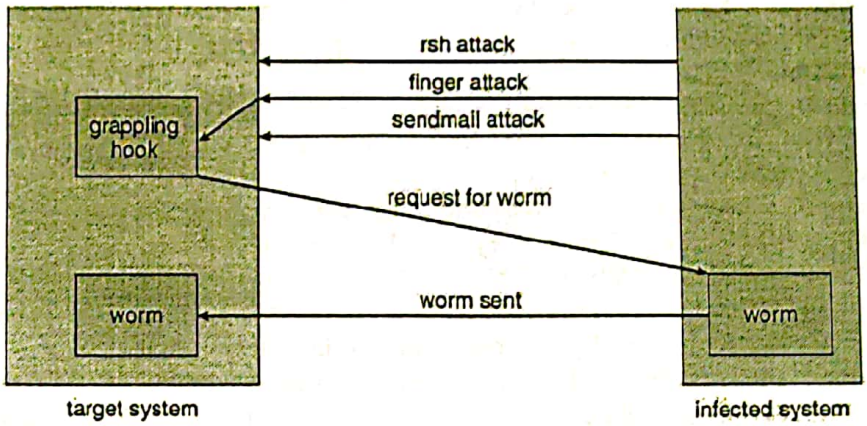
- ✓ Worms
- ✓ Port Scanning
- ✓ Denial of Service

I. Worms

A worm is a process that uses the fork / spawn process to make copies of itself in order to create havoc on a system. Worms consume system resources, often blocking out other, legitimate processes. Worms that propagate over networks can be especially problematic, as they can tie up vast amounts of network resources and bring down large-scale systems. One of the most well-known worms was launched by Robert Morris, calles as Morris worm .This worm consisted of two parts:

1. A small program called a ***grappling hook,*** which was deposited on the target system through one of three vulnerabilities, and
2. The main worm program, which was transferred onto the target system and launched by the grappling hook program.

205/214



The three vulnerabilities exploited by the Morris Internet worm were as follows:

1. **rsh (remote shell)** is a utility that was in common use at that time for accessing remote systems without having to provide a password.
2. **finger** is a utility that allows one to remotely query a user database, to find the true name and other information for a given account name on a given system.
3. **sendmail** is a routine for sending and forwarding mail that also included a debugging option for verifying and testing the system

II. Port Scanning

Port Scanning is technically not an attack, but rather a search for vulnerabilities to attack. The basic idea is to systematically attempt to connect to every known (or common or possible) network port on some remote machine, and to attempt to make contact.
Once it is determined that a particular computer is listening to a particular port, then the next step is to determine what daemon is listening, and whether or not it is a version containing a known security flaw that can be exploited.
Because port scanning is easily detected and traced, it is usually launched from ***zombie systems,*** i.e. previously hacked systems that are being used without the knowledge or permission of their rightful owner.

III. Denial of Service

Denial of Service (DOS) attacks do not attempt to actually access or damage systems, but merely to clog them up so badly that they cannot be used for any useful work. Tight loops that repeatedly request system services are an obvious form of this attack.
DOS attacks can also involve social engineering, such as the Internet chain letters that say "send this immediately to 10 of your friends, and then go to a certain URL", which clogs up not only the Internet mail system but also the web server to which everyone is directed. (Note: Sending a "reply all" to such a message notifying everyone that it was just a hoax also clogs up the Internet mail service, just as effectively as if you had forwarded the thing.)

Security systems that lock accounts after a certain number of failed login attempts are subject to DOS attacks which repeatedly attempt logins to all accounts with invalid passwords strictly in order to lock up all accounts.

➢ **Cryptography as a Security Tool**

Within a given computer the transmittal of messages is safe, reliable and secure, because the OS knows exactly where each one is coming from and where it is going.
On a network, however, things aren't so straightforward - A rogue computer (or e-mail sender) may spoof their identity, and outgoing packets are delivered to a lot of other computers besides their (intended) final destination, which brings up two big questions of security:

- **Trust** - How can the system be sure that the messages received are really from the source that they say they are, and can that source be trusted?
- **Confidentiality** - How can one ensure that the messages one is sending are received only by the intended recipient?

Cryptography can help with both of these problems, through a system of **secrets** and **keys.**
Keys are designed so that they cannot be divined from any public information, and must be guarded carefully.

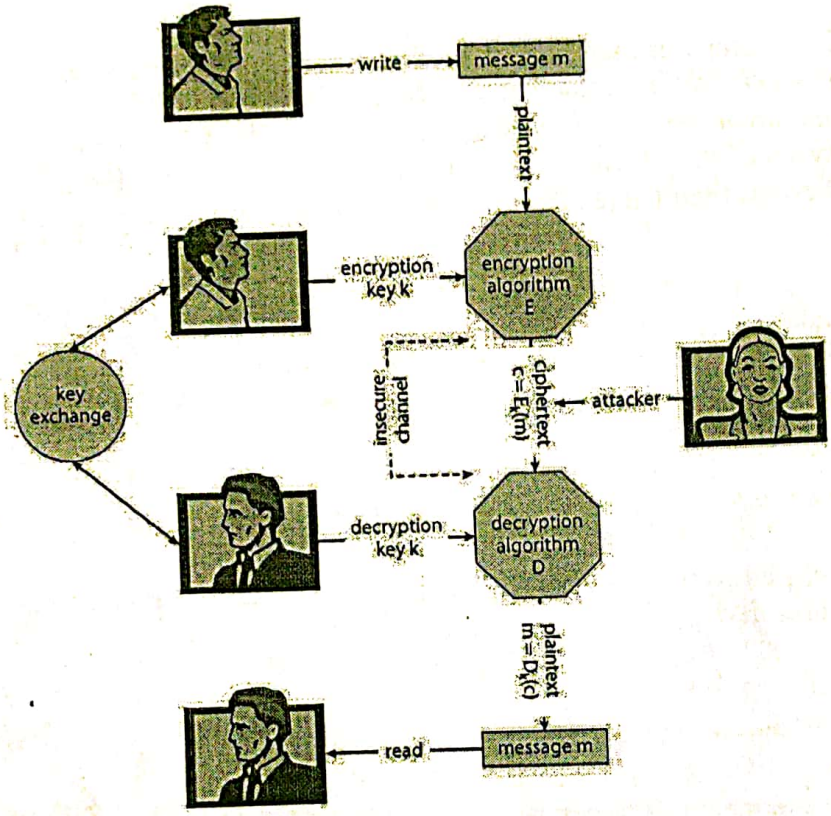
- ✓ Encryption
- ✓ Authentication in cryptography
- ✓ Key Distribution
- ✓ Implementation of Cryptography

I.Encryption

The basic idea of encryption is to encode a message so that only the desired recipient can decode and read it. Encryption has been around since before the days of Caesar, and is an entire field of study in itself. Only some of the more significant computer encryption schemes will be covered here.
The basic process of encryption is shown in Figure. The steps in the procedure and some of the key terminology are as follows:

1. The **sender** first creates a **message, m** in plaintext.
2. The message is then entered into an **encryption algorithm, E,** along with the **encryption key, Ke.**
3. The encryption algorithm generates the **ciphertext, c, = E(Ke)(m).** For any key k, E(k) is an algorithm for generating ciphertext from a message, and both E and E(k) should be efficiently computable functions.
4. The ciphertext can then be sent over an unsecure network, where it may be received by **attackers.**
5. The **recipient** enters the ciphertext into a **decryption algorithm, D,** along with the **decryption key, Kd.**

6. The decryption algorithm re-generates the plaintext message, m, = D(Kd)(c). For any key k, D(k) is an algorithm for generating a clear text message from a ciphertext, and both D and D(k) should be efficiently computable functions.



There are 2 types of encryption

i. Symmetric encryption
ii. Asymmetric encryption

Symmetric Encryption

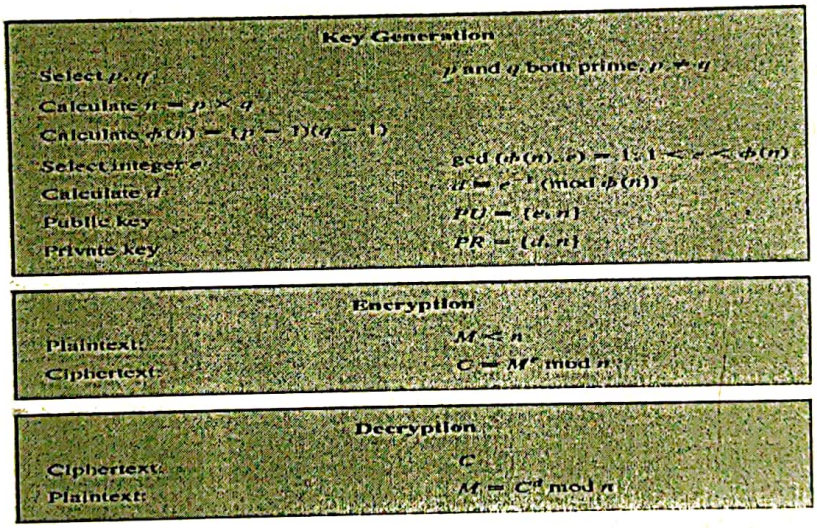
With ***symmetric encryption*** the same key is used for both encryption and decryption, and must be safely guarded. There are a number of well-known symmetric encryption algorithms that have been used for computer security:

The ***Data-Encryption Standard, DES*** is known as a ***block cipher,*** because it works on blocks of data at a time. Messages are broken down into 64-bit chunks, each of which is encrypted using a 56-bit key through a series of substitutions and transformations.

The ***Advanced Encryption Standard, AES,*** developed by NIST in 2001 to replace DES uses key lengths of 128, 192, or 256 bits, and encrypts in blocks of 128 bits using 10 to 14 rounds of transformations on a matrix formed from the block.
The ***two fish algorithm,*** uses variable key lengths up to 256 bits and works on 128 bit blocks.
RC5 can vary in key length, block size, and the number of transformations, and runs on a wide variety of CPUs using only basic computations.
RC4 is a ***stream cipher,*** meaning it acts on a stream of data rather than blocks. The key is used to seed a pseudo-random number generator, which generates a ***key stream*** of keys.

Asymmetric Encryption

With ***asymmetric encryption,*** the decryption key, Kd, is not the same as the encryption key, Ke, and more importantly cannot be derived from it, which means the encryption key can be made publicly available, and only the decryption key needs to be kept secret. One of the most widely used asymmetric encryption algorithms is ***RSA,*** named after its developers - Rivest, Shamir, and Adleman., which is given as follows:



II. *Authentication in cryptography*

Authentication involves verifying the identity of the entity that transmitted a message.

The following are few authentication approaches used in cryptography

Hash functions, H(m) generate a small fixed-size block of data known as a ***message digest,*** or ***hash value*** from any given input data. Popular hash functions are **MD5,** which generates a 128-bit message digest, and **SHA-1,** which generates a 160-bit digest.

209/24

Message digests are useful for detecting (accidentally) changed messages, but are not useful as authenticators

A ***message-authentication code, MAC,*** uses symmetric encryption and decryption of the message digest, which means that anyone capable of verifying an incoming message could also generate a new message.

An asymmetric approach is the ***digital-signature algorithm,*** which produces authenticators called ***digital signatures***

III. *Key Distribution*

Key distribution with symmetric cryptography is a major problem, because all keys must be kept secret, and they obviously can't be transmitted over unsecure channels. One option is to send them ***out-of-band,*** say via paper or a confidential conversation.

Another problem with symmetric keys, is that a separate key must be maintained and used for each correspondent with whom one wishes to exchange confidential information. Asymmetric encryption solves some of these problems, because the public key can be freely transmitted through any channel, and the private key doesn't need to be transmitted anywhere. Recipients only need to maintain one private key for all incoming messages, though senders must maintain a separate public key for each recipient to which they might wish to send a message. Fortunately the public keys are not confidential, so this ***key-ring*** can be easily stored and managed

One solution to the above problem involves ***digital certificates,*** which are public keys that have been digitally signed by a trusted third party

IV. *Implementation of Cryptography*

Network communications are implemented in multiple layers - Physical, Data Link, Network, Transport, and Application being the most common breakdown.

Encryption and security can be implemented at any layer in the stack, with pros and cons to each choice:

- Because packets at lower levels contain the contents of higher layers, encryption at lower layers automatically encrypts higher layer information at the same time.
- However security and authorization may be important to higher levels independent of the underlying transport mechanism or route taken.

At the network layer the most common standard is **IPSec,** a secure form of the IP layer, which is used to set up **Virtual Private Networks, VPNs.**

At the transport layer the most common implementation is SSL

210/214

User Authentication

User authentication is a process that allows a device to verify the identify of someone who connects to a network resource

- ✓ *Passwords*
- ✓ *Encrypted Passwords*
- ✓ *One-Time Passwords*
- ✓ *Biometrics*

I.Passwords

Passwords are the most common form of user authentication. If the user is in possession of the correct password, then they are considered to have identified themselves.
In theory separate passwords could be implemented for separate activities, such as reading this file, writing that file, etc.

Password Vulnerabilities

Passwords can be guessed
"Shoulder surfing" involves looking over people's shoulders while they are typing in their password.
"Packet sniffing" involves putting a monitor on a network connection and reading data contained in those packets.
Most systems have configurable parameters controlling password generation and what constitutes acceptable passwords.

II. Encrypted Passwords

Modern systems do not store passwords in clear-text form, and hence there is no mechanism to look up an existing password.
Rather they are encrypted and stored in that form. When a user enters their password, that too is encrypted, and if the encrypted version match, then user authentication passes.

III. One-Time Passwords

These are often based on a **challenge** and a **response**. Because the challenge is different each time, the old response will not be valid for future challenges.

IV. Biometrics

Biometrics involve a physical characteristic of the user that is not easily forged or duplicated and not likely to be identical between multiple users.

Fingerprint scanners are getting faster, more accurate, and more economical.
Palm readers can check thermal properties, finger length, etc.
Retinal scanners examine the back of the users' eyes.
Voiceprint analyzers distinguish particular voices.

211/214

Implementing Security Defenses

- ✓ Auditing, Accounting, and Logging
- ✓ Virus Protection
- ✓ Intrusion Detection
- ✓ Vulnerability Assessment
- ✓ Security Policy

I.Security Policy

A security policy should be well thought-out, agreed upon, and contained in a living document that everyone adheres to and is updated as needed.
Examples of contents include how often port scans are run, password requirements, virus detectors, etc.

II. Vulnerability Assessment

Periodically examine the system to detect vulnerabilities.

- Port scanning.
- Check for bad passwords.
- Look for suid programs.
- Unauthorized programs in system directories.
- Incorrect permission bits set.

III.Intrusion Detection

Intrusion detection attempts to detect attacks, both successful and unsuccessful attempts. Different techniques vary along several axes:

- The time that detection occurs, either during the attack or after the fact.
- The types of information examined to detect the attack(s). Some attacks can only be detected by analyzing multiple sources of information.
- The response to the attack, which may range from alerting an administrator to automatically stopping the attack (e.g. killing an offending process), to tracing back the attack in order to identify the attacker.

Another approach is to divert the attacker to a ***honeypot***, on a ***honeynet.*** The idea behind a honeypot is a computer running normal services, but which no one uses to do any real work. Such a system should not see any network traffic under normal conditions, so any traffic going to or from such a system is by definition suspicious.

There are two major approaches to detecting problems:

- ***Signature-Based Detection*** scans network packets, system files, etc. looking for recognizable characteristics of known attacks, such as text strings for messages or the binary code for "exec /bin/sh".
- ***Anomaly Detection*** looks for "unusual" patterns of traffic or operation, such as unusually heavy load or an unusual number of logins late at night.

IV. *Virus Protection*

Modern anti-virus programs are basically signature-based detection systems, which also have the ability (in some cases) of ***disinfecting*** the affected files and returning them back to their original condition.

Both viruses and anti-virus programs are rapidly evolving. For example viruses now commonly mutate every time they propagate, and so anti-virus programs look for families of related signatures rather than specific ones.

V. *Auditing, Accounting, and Logging*

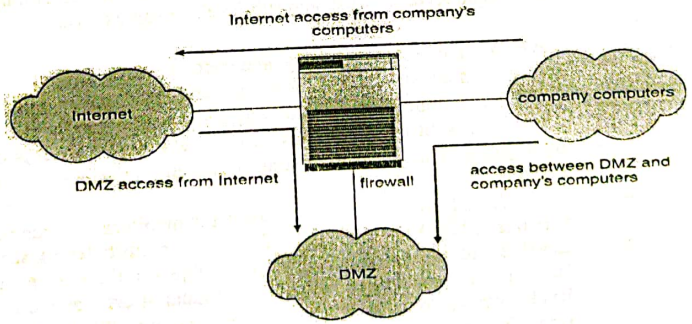
Auditing, accounting, and logging records can also be used to detect anomalous behavior. "The Cuckoo's Egg" tells the story of how Cliff Stoll detected one of the early UNIX break ins when he noticed anomalies in the accounting records on a computer system being used by physics researchers.

Firewalling to Protect Systems and Networks

Firewalls are devices (or sometimes software) that sit on the border between two security domains and monitor/log activity between them, sometimes restricting the traffic that can pass between them based on certain criteria.

For example a firewall router may allow HTTP: requests to pass through to a web server inside a company domain while not allowing telnet, or other traffic to pass through.

A common architecture is to establish a de-militarized zone, DMZ, which sort of sits "between" the company domain and the outside world, as shown below. Company computers can reach either the DMZ or the outside world, but outside computers can only reach the DMZ. Perhaps most importantly, the DMZ cannot reach any of the other company computers, so even if the DMZ is breached, the attacker cannot get to the rest of the company network.



Firewalls themselves need to be resistant to attacks, and unfortunately have several vulnerabilities:

Tunneling, which involves encapsulating forbidden traffic inside of packets that are allowed

Denial of service attacks addressed at the firewall itself.

Spoofing, in which an unauthorized host sends packets to the firewall with the return address of an authorized host.

In addition to the common firewalls protecting a company internal network from the outside world, there are also some specialized forms of firewalls that have been recently developed:

A ***personal firewall*** is a software layer that protects an individual computer. It may be a part of the operating system or a separate software package.

An ***application proxy firewall*** understands the protocols of a particular service and acts as a stand-in (and relay) for the particular service.

XML firewalls examine XML packets only, and reject ill-formed packets. Similar firewalls exist for other specific protocols.

System call firewalls guard the boundary between user mode and system mode, and reject any system calls that violate security policies.

Computer-Security Classifications

No computer system can be 100% secure, and attempts to make it so can quickly make it unusable.

However one can establish a level of trust to which one feels "safe" using a given computer system for particular security needs.

The U.S. Department of Defense's "Trusted Computer System Evaluation Criteria" defines four broad levels of trust, and sub-levels in some cases:

a. Level D is the least trustworthy, and encompasses all systems that do not meet any of the more stringent criteria. DOS and Windows 3.1 fall into level D.

b. Level C1 includes user identification and authorization, and some means of controlling what users are allowed to access what files. It is designed for use by a group of mostly cooperating users, and describes most common UNIX systems.

c. Level C2 adds individual-level control and monitoring. Some special secure versions of UNIX have been certified for C2 security levels, such as SCO.

d. Level B adds sensitivity labels on each object in the system, such as "secret", "top secret", and "confidential". Individual users have different clearance levels, which controls which objects they are able to access. All human-readable documents are labeled at both the top and bottom with the sensitivity level of the file.

e. Level B2 extends sensitivity labels to all system resources, including devices. B2 also supports covert channels and the auditing of events that could exploit covert channels.

f. B3 allows creation of access-control lists that denote users NOT given access to specific objects.

g. Class A is the highest level of security. Architecturally it is the same as B3, but it is developed using formal methods which can be used to ***prove*** that the system meets all requirements and cannot have any possible bugs or other vulnerabilities. Systems in class A and higher may be developed by trusted personnel in secure facilities.

h. **These classifications determine what a system *can* implement, but it is up to security policy to determine *how* they are implemented in practice.**

These systems and policies can be reviewed and certified by trusted organizations, such as the National Computer Security Center. Other standards may dictate physical protections and other issues.