(DEEMED TO BE UNIVERSITY)

Cyber Security and Blockchain Technology (CSBT)

Network And Infrastructure Security

CSE Hounors

Dr.Yogesh Kumar Sharma
Ramaiah Challa
Dr.A.Roshini
D.Sehkar Babu

Cohort E09 2023

www.kluniversity.in

NETWORK AND INFRASTRUCTURE SECURITY

Dr Yogesh Kumar Sharma, Ramaiah Challa,
Dr Roshini A, Sekhar Babu D.

NETWORK AND INFRASTRUCTURE SECURITY

ISBN: XXXXXXXXXXXXX

Printed in the USA by 48 Hour Books (www.48HrBooks.com)

DEDICATION

This book is dedicated to our loving family members of all the authors, whose unwavering support and encouragement have been the cornerstone of my life and my writing journey. Your love, belief in our abilities, and countless acts of kindness have inspired me to pursue my dreams with passion and determination.

This book is also dedicated to the Management of K. L. E. F for their visionary leadership and unwavering support in fostering a nurturing environment of learning and innovation.

Phone 9966302375
Email: ramaiah.challa@kluniversity.in
Online : https:kluniversity.in

TABLE OF CONTENTS

INTRODUCTION

The main objective of this book is to make the students understand the various techniques to device security solutions for network infrastructure. Students will be able to explore their knowledge in directory services and Hardware procedures for acquiring digital certificate. This course provides a clear insight on different security solutions towards E-mail and Web Security issues, in turn students will be able to understand the significance of the security in real time. This course enables students to identify vulnerabilities in the context of network infrastructure security.

Phone 9966302375.
Email: ramaiah.challa@kluniversity.in

Chapter One - Introduction to Network Infrastructure Security

1.1 Network Hardware, Software and Services, Hardware Protection

1.1.1 Network Infrastructure Security

In business IT settings, the term "Network Infrastructure Security" refers to the practice of securing the underlying networking infrastructure against threats such as unauthorized access, modification, deletion, and theft.

Access control, application security, firewalls, VPNs, behavioral analytics, intrusion prevention systems, and wireless security are all examples of possible security methods.

1.1.2 Hardware for a Network

The term "network hardware" refers to the collection of physical or network devices required for hardware units on a computer network to interact and communicate with one another.

These are the specialized pieces of hardware that link together to make a network run smoothly and efficiently.

Hardware Categories for Networks

Hardware for a network can come in a variety of forms. Those are

- Modem
- Router
- Hubs
- Bridges
- Switches
- Network Interface Cards
- Network Cables
- Firewall

Modem

With the use of a modem, a computer may access the World Wide Web using a regular phone connection. M/D refers to a modulator and a demodulator. Through this method, digital information may be sent via analog telephone lines. At one end, a modem takes the digital data from the computer and transforms them to analog signals that can be transmitted over a telephone line. The output digitalizes the analog signals so that they may be read by another computer.

A modem may be broken down into the following types based on its speed and data transfer capacity:

• A dial-up or standard computer modem

Telecommunications Modem for Mobile Phones

Cable modem

Routers

A router is a piece of hardware that bridges a local area network (LAN) to the wider internet. It takes in incoming packets, examines them, and then sends them on to another network.

A router may join many networks together. The router is often used to link a local area network (LAN) at home or in the business to a wide area network (WAN). In most cases, it will also include cables for connecting PCs on a local area network.

Wireless (Wi-Fi) LAN connections are another option for making a network equipment mobile. These are also known as WAPs, which stands for wireless access points.

If the router has the necessary information in its routing database, it will send the packet on its way.

• It selects the optimal path for packet transfer from among all possible ones.

• In the OSI Reference model, a router operates at the Network layer.

Hub

A hardware device called a "Hub" shares a single network connection among several other gadgets.

• It sends out a signal to every computer in a network. A computer will initially submit a request to the Hub via cable when it needs data from a network. This request will be published by Hub to all nodes in the network.

• Every gadget will verify that the request is meant for it. In such case, your request won't be processed.

• As a result, it uses a lot of data transfer because some computers may not need to get the broadcasted information. In a local multiplayer game, the hub might connect a few game consoles across a connected or wireless LAN.

• Hubs are no longer used since more sophisticated computer network components like switches and routers have replaced them.

Bridge

Two local area networks can be joined via a bridge. Before delivering a signal, it looks around for a compatible device.

• This suggests that it doesn't send data if the target device isn't nearby. In addition, it verifies that the destination device has not already received the message.

The network's overall efficiency is boosted thanks to these measures.

Switch

• A Switch is superior than a hub or bridge, yet it serves a same purpose.

When the demand is strong, a switch's efficiency increases because it decreases the amount of delay by storing the MAC

addresses of network devices and transferring data packets only to those devices that have requested them.

Network Interface Cards

• A network interface card (NIC) is a piece of hardware installed on a computer that facilitates communication between computers on a network. It usually takes the shape of a chip or circuit board. While most current computers have NICs integrated directly onto the motherboard, others require the addition of a separate expansion card in the form of a tiny circuit board.

• It's compatible with data rates between 10,100 and 1000 Mb/s.

• The IEEE assigns each network interface card a unique identifier called a media access control (MAC) address, which is stored in binary form on the network card's embedded microchip. The PROM stores the MAC address.

• Wired and wireless NICs are also available.

Wired LAN support is built into the motherboard, so yes! With a wired NIC, data is sent through physical connections like cables and plugs.

The antenna needed to connect to a wireless network is housed in the wireless NIC. The wireless NIC is typically built into laptops.

Network Cables & Firewall

Cables:

The many components of a network are linked through cables. A signal can be sent through a cable, which is a type of transmission medium.

There are three distinct kinds of transmission cables:

TPC (twisted pair cable)

Cable, coaxial

Optical fiber

Cable connections are preferred over wireless ones in modern networks because they are more secure, less vulnerable to assaults, and can handle more data in a given amount of time.

Firewall:

A firewall is a piece of equipment, either hardware or software, that stands between a computer and the rest of the network. In order to prevent unauthorized access to a local area network (LAN), a firewall is typically installed between the LAN and the internet.

A firewall is a security system that lets approved traffic through (such as emails or web pages) but prevents unwanted visitors from accessing a private network.

1.1.3 Network Software

• Networking systems cannot function without network software. It aids network administrators and security officers in

simplifying network management and traffic monitoring and control.

• Network software is vital to the success of any network because it streamlines and improves communication, security, content, and data sharing.

Software for managing, maintaining, and monitoring computer networks falls under the umbrella term "network software."

• Network software may range from simple tools like network and protocol analyzers to comprehensive suites that do everything from secure and monitor networks to map them and assess their performance to keep track of their assets.

1.1.4 Types of Network Software

Some common types of network software are:

Tools for monitoring network activity, such as traffic volume, speed, and packet loss, are invaluable to IT staff. Nagios, SolarWinds Network Performance Monitor, PRTG Network Monitor, and Zabbix are just a few examples of well-known network monitoring software.

Software for network defense aims to identify and neutralize threats to a network's security, such as malicious code, viruses, and intrusions. Firewalls, intrusion detection systems, endpoint protection software, and virtual private network software are all examples of common types of network security software.

It is possible to view the network's design, comprehend its performance, and spot any bottlenecks or vulnerabilities with the help of network mapping and visualization tools. SolarWinds Network Topology Mapper and Microsoft Visio are two examples of network mapping tools.

Software for managing and maintaining an inventory of network hardware, software, and other assets is essential for IT departments. Network inventory tools like Spiceworks and ManageEngine AssetExplorer are two examples.

Networked users can easily exchange data and other resources thanks to file sharing software. BitTorrent, Dropbox, and Google Drive are all examples of file-sharing software.

Connecting to a network from afar is made possible by remote access software. VPN clients, remote desktop applications, and terminal emulation programs are all types of remote access software.

Internet users may view and interact with webpages on the world wide web thanks to software called web browsers. Google Chrome, Mozilla Firefox, and Microsoft Edge are all examples of browsers.

Operating systems: software that controls the use of a computer's hardware and offers standard services to other applications.

1.1.5 Data Transfer over Networks

Services supplied by a network that allow for the exchange of information and the connection of devices are known as "network services."

The following are some examples of common network services:

In order to access and share files on various devices, users can take advantage of file sharing services, which allow users to exchange files and resources across a network.

Email: Email services enable users to transmit and receive electronic mail and data across a network, facilitating quick and easy interaction with others.

Printing: With printing services, customers may share printers over a network so that several computers can print to the same device.

To gain access to a network's resources and services while physically located in a different place, users can take use of remote access services.

Web hosting is a service that allows individuals and businesses to store their websites and other web-based applications on a server so that they may be accessed over the World Wide Web.

Voice and video conferencing: These services allow users to have meetings and interact with people remotely by communicating in real-time via a network using voice and/or video.

The Domain Name System (DNS) is a service that helps people utilize the Internet by converting domain names to numerical IP addresses.

Dynamic Host Configuration Protocol (DHCP) is a network service that simplifies the process of managing and configuring network devices by allocating IP addresses to them automatically.

Network Time Protocol (NTP) is a service that ensures all devices on a network are using the same time reference by synchronizing their internal clocks.

Network services allow users to connect with one another, share information, and have access to various digital resources and services.

1.1.6 How to Secure Hardware/ Software Protection

Hardware security refers to the precautions taken to safeguard computers and networks from physical threats including theft, vandalism, and intrusion. Some instances of hardware security include the following:

- Padlocks and cable locks, for example, can be used to keep thieves out of the cases of computers, laptops, and servers.

- Security cameras are a useful tool for keeping tabs on the locations that house servers and other network hardware.

- Physical access to computers and networks can be limited by using biometric authentication technology like fingerprint scanners and face recognition systems.

- Firewalls are network security gear that may be set up to prevent unwanted traffic, viruses, and other risks from entering a network.

- In the event of a blackout or power surge, hardware components can be safeguarded by using a power backup system, such as an Uninterruptible Power Supply (UPS).

- Environmental controls: Computer and network equipment may be shielded from heat, moisture, and dust by using

temperature and humidity sensors, air conditioning systems, and ventilation fans.

- Encryption: Data saved on hardware components like hard drives or USB devices can be protected against unwanted access with the use of encryption technologies in the event that the equipment is lost or stolen.

Protecting computer and network gear against things like theft, damage, and intrusion is crucial to ensuring their continued operation and security.

1.2 Intruders and Virus

1.2.1 Intruders

When discussing the topic of computer security, an intruder is defined as anybody or anything that breaches the protections of a computer, network, or other electronic device. People who break in might be motivated by anything from mere curiosity to malice. In order to steal information, disrupt services, or engage in other malicious actions, hackers might take advantage of flaws in software, networks, or even user behavior.

Hackers, cybercriminals, malevolent insiders, and automated scripts or bots are only few examples of intruders. To bypass protections and enter inappropriately, they can employ a wide range of strategies and instruments.

Intruders frequently employ techniques like as:

Through "social engineering," hackers coerce victims into giving over private information or granting them access to computer networks. Email phishing, pretexting, and impersonation are all methods that can be used for this purpose.

Hackers will often try to guess or crack passwords in order to gain access to protected resources. Attack methods range from simple guesses to more complex strategies like rainbow tables.

Attacks on a network happen when hackers use such networks' flaws to get in. Scanning networks, scanning ports, and exploiting vulnerabilities in network protocols are all examples of this.

Malware: Hackers can employ viruses, worms, and Trojan horses to access systems, steal data, and destroy networks. Malware can be spread by infected attachments in emails, rogue websites, or hacked applications.

Security flaws in computer programs or operating systems are often exploited by hackers so that they can get unauthorized access. They may exploit computers that are running old or unpatched software.

When a system is breached, the results might be very different depending on the intruder's goals and the compromised system. The damage caused by a cyberattack can vary widely, from the theft or loss of private information to monetary losses,

service interruptions, and even the compromising of essential infrastructure.

Passwords, software updates, firewalls, intrusion detection systems, and antivirus software are just some of the important security measures that must be put in place to ward against hackers. Understanding and protecting yourself from social engineering assaults requires user education and awareness.

1.2.2 Virus:

In the context of computer security, a virus is any piece of malware that may copy itself and propagate to other devices over the internet. Viruses are malicious software that infect computers by copying themselves onto other files or applications and then wreaking havoc or stealing private data.

Some fundamental facts concerning viruses are as follows:

Viruses may spread from one computer to another by copying their code and inserting it into other files. The virus is activated and begins spreading to other files or systems when the infected software or file is executed or accessed, typically without the user's knowledge or agreement. Email attachments, compromised websites, shared resources on the network, and portable media are only few of the vectors for virus distribution.

Viruses often have a predetermined payload, or the harmful action they take when triggered. The virus's payload can be tailored to its intended purpose.

Common behaviors of viruses include:

The corruption of data occurs when a virus replaces or deletes a file.

By opening backdoors or introducing vulnerabilities, viruses can grant illegal access to the system they infect.

Theft of Personal Information Some viruses are programmed specifically to steal login credentials, financial information, or other private details.

Infection with a virus can lead to a computer being a member of a botnet, which is a network of hacked computers used for nefarious reasons like transmitting spam or executing distributed denial of service attacks.

Files on an infected machine are encrypted and a ransom is demanded in exchange for the decryption key. This form of virus is known as ransomware.

Virus Types:

Viruses come in a wide variety, and each has its own quirks and ways of spreading. Here are some of the most frequent:

- Viruses that infect files do so by affixing themselves to text files, spreadsheets, or executable programs.

- Viruses known as macros target and infect documents created in Microsoft Office and other programs that employ macros.
- Viruses that infect the boot sector of storage media such as hard drives or floppy disks are known as boot sector viruses.
- Polymorphic viruses are more difficult to identify and eradicate because they can alter their coding or signature.

Although not viruses, worms are a type of malicious software capable of replicating themselves and spreading over networks without the help of a host application.

Trojan Horses are malicious programs that masquerade as safe files or programs but actually contain harmful code.

Security And Avoidance:

Defending against viruses calls for a tiered defense system:

Install a reliable antivirus program that scans for and removes viruses and other forms of malware. Always use the most recent version of the antivirus program for maximum protection.

Viruses can exploit security flaws in your operating system and other apps if you don't update them on a regular basis.

Use anti-virus software and web filtering software to prevent your computer from being infected with malware.

Practice safe browsing behavior by being wary of downloading files or clicking on links. Steer clear of questionable or unreliable sources.

System backups: copy your data on a regular basis to an offsite location, such as a hard drive or the cloud. Your files can be restored from a safe backup in the event of a virus infestation.

1.3 Network Management Security, Device Security & VPN

1.3.1 Network Management and Security

Control and monitoring are essential for any system of any complexity. Device deployment, integration, and coordination were all part of network management's purview.

- Monitor
- Test
- Configure
- Analyze
- Evaluate & Manage

the network and all of its parts.

The goal of network administration is to ensure that the needs of the network are met at an affordable price.

Availability

Efficiency in business operations

Excellent service for a fair price.

The network is quite diverse. Standards are necessary for devices to exchange data, and share information guidelines for Network Administration

- SNMP stands for "Simple Network Management Protocol."
- Create using the client-server model.
- Polling-based voting system.
- Guidelines for managing networks in practice.
- Web-based administration and portability between platforms.

Introduction to SNMP

Management Information Base (MIB)

A Data Warehouse for Network Administration

SMI stands for "Management Information Structure."

Language for defining data on MIB objects

The SNMP protocol

- Protocol for communication, directives
- Safeguarding features and control options
- Security concerns were resolved in SNMPv3, which also served as a foundation for future SNMP releases.

SMI & MIB

SMI allows for the definition of managed objects and the behavior of such objects. To define managed objects (MIB), SNMP relies on its data definition language, SMI.

Object definitions (in SMI syntax) are provided by MIB. The information is more tailored to individual providers. (MIB-II, RFC 1213). The agent transmits data from the MIB or makes adjustments per the instructions of a remote management.

Each managed resource has its own Management Information Base (MIB), which details the resource's exposed interfaces. For instance, a server's MIB may detail its central processing unit (CPU), memory system, and network interface protocols, while a router's MIB may detail its network interface speeds.

1.3.2 Device Security

Devices in the network's architecture facilitate the transfer of data, applications, services, and multimedia. Among these tools are

Hardware used in computer networks, including gateways, firewalls, switches, servers, load balancers, IDS/IPS, DNS, and SANs.

Most or all company and consumer traffic must go via these devices, making them prime targets for malevolent cyber attackers.

If an attacker gains access to a company's gateway router, they will be able to snoop on, alter, and even block outgoing communications.

Someone who has compromised internal networks by gaining access to routing and switching devices.

Key hosts' traffic can be monitored, modified, or blocked entirely.

Utilize dependable connections to hop from one host to another.

Insecure network infrastructure components include:

The devices that make up a network's infrastructure are a soft target for hackers.

Unlike standard PCs and servers, many network devices aren't kept as secure after installation.

Other potential causes of network device vulnerability include:

Antivirus software, integrity maintenance, and other security measures that assist safeguard general-purpose hosts are rarely installed on residential and small office/home office routers.

These exploitable services are built into and distributed by the manufacturers of these network devices, which also facilitates their setup, operation, and upkeep.

Network administrators frequently don't alter vulnerable factory settings, make their equipment more robust for use, or keep them up-to-date with security patches.

Once a customer's equipment is no longer maintained by the manufacturer or seller, the ISP is not obligated to replace it.

When conducting post-intrusion investigations, searching for attackers, and restoring general-purpose hosts, owners and operators frequently fail to inspect network devices.

Separate Networks and Their Purposes

Network segmentation done right is a powerful security measure that may

The spread of exploits or lateral movement by an intruder inside a network.

Segregation divides networks into subsystems that perform certain functions.

The Physical Isolation of Private Data

Network segments should be designed using the concepts of least privilege and need-to-know.

Segment your network to protect private data and other sensitive information.

Secure all network nodes and protocols by following best practices and settings.

Safeguarding Private Data Through Virtual Segmentation

Isolate a user from the public network by creating a separate VLAN for them.

To divide network traffic among numerous routing tables on a single router, implement virtual routing and forwarding (VRF) technology.

Virtual Private Networks (VPNs) allow you to expand a private or public network securely.

Reduce Side-to-Side Chatter That isn't Necessary

Host-based firewall rules can be used to stifle network traffic coming from other hosts.

Use a filter to manage who may enter and leave a VLAN, known as a VLAN access control list (VACL).

Administrators can partition the network logically, either physically or virtually, to keep track of sensitive data on separate networks.

Secure Network Equipment

Turn off any unsecured remote administration protocols (such as Telnet or FTP) that are in use for managing the network.

Services like discovery protocols and source routing can be turned off if they aren't being used.

Make use of SNMP version 3 (or later).

Safeguarding connection to the virtual terminal, auxiliary lines, and console.

Use the most secure password encryption method and a strict password policy.

To secure routers and switches, it is important to manage remote access.

Limit who can go in close proximity to your routers and switches. Keep a copy of your settings in a non-online location.

Safely Access Critical Infrastructure Equipment

Users can be granted administrative privileges to view restricted materials.

Since intruders might take advantage of administrative rights, restricting them is vital to the security of infrastructure equipment.

Put in place MFA (multi-factor authentication).

User-revealed information (such as a password),

Something the user physically has (a token, for instance) and

An individual's own identifying feature (like a fingerprint).

Control who can do what.

Access information for managing network devices should be kept in a server that offers authentication, authorization, and accounting (AAA) services.

Take charge of administrator logins.

Default passwords should be changed.

Password guidelines for security

Password storage should always use salted hashes.

Passwords should be maintained in a safe, off-network area for quick access in an emergency.

Carry out Management Outside the Box

OoB management allows for the remote administration of network infrastructure components using alternative communication channels.

These isolated channels of communication can be set up in a variety of ways, from virtual tunneling to complete physical isolation.

In OoB management, the physical and digital worlds can coexist, or a combination of the two can be used.

Normal network traffic should be isolated from management data.

Make sure that all device management traffic originates from Out of Band.

Encrypt all channels used for administration.

Remote connections to critical infrastructure components like terminal and dial-in servers should be encrypted.

All administrative tasks should be handled from a single, completely patched host through an encrypted connection, ideally on OoB.

Check the Hardware and Software to Ensure Its Security

Counterfeit, secondary market, and gray market products all refer to the same thing: goods obtained in an illegal manner.

Users' data and the security of the network as a whole are at considerable danger from unauthorized devices and software.

Always know who you're buying from and make sure they're allowed to offer your product.

Make distributors and retailers do supply-chain integrity checks to ensure legitimate hardware and software. After setup, make sure no gadgets have been tampered with.

Use a variety of methods to verify serial numbers.

Get your apps, fixes, and upgrades from reputable sites.

Verify the firmware using a hash and check its values against the vendor's database to spot any tampering.

Maintain a program of routine monitoring and logging of devices, checking their network settings.

Owners, administrators, and buyers of networks should get training on how to spot gray market hardware.

1.3.3 VPN or Virtual Private Networks

Security on the Internet and large-scale WANs is an area where virtual private networks (VPNs) have shown great potential.

Commercial and military suppliers alike have come to rely on virtual private networks (VPNs) as a safe and secure means of connecting to the Internet and among themselves.

It's easy to set up a VPN, it's quite safe, and it doesn't cost anything to run. When it came to securing their networks, many banks considered VPNs to be the best alternative.

VPNs are used to establish encrypted connections between offices and other network nodes.

Virtual private networks provide encrypted tunnels across public networks like the Internet, protecting sensitive data in transit.

Concerns regarding data security have recently been raised by companies using the Internet to set up virtual private networks (VPNs).

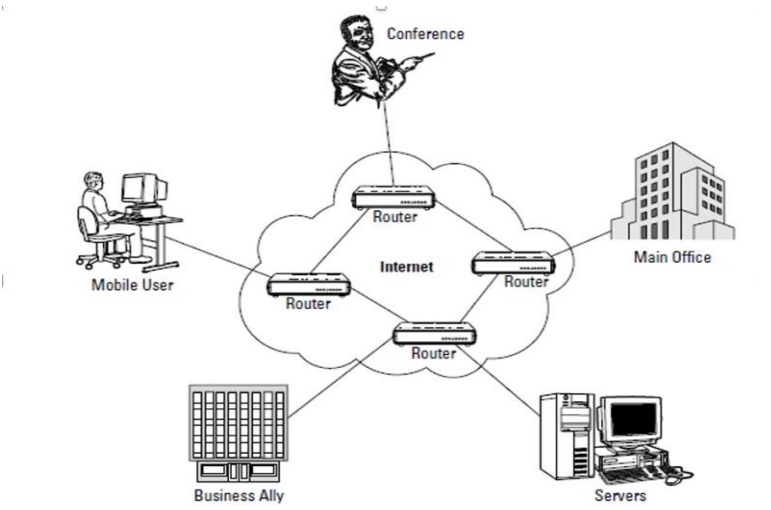

Figure1.1 In this example, we will look at a typical VPN connection with two distinct final domains. (Copy right is reserved to William Stallings.)

1.4 Data Center & Enterprise Networks Security, Securing A Wireless Lan

1.4.1 Data Center Security (Safety in Data Centers)

How do you define data center safety

The term "data center security" refers to the procedures and precautions used to prevent breaches in security at a data center.

Data center security encompasses both physical and network layers, with the former needing careful site planning to restrict physical break-ins and the latter requiring the installation of firewalls and anti-malware software by trained security professionals.

Without a doubt, no one.

Having several physical and network security measures in place is essential for any company that runs all or part of its activities out of a data center.

In order to prevent the data stored in the data center from being lost, tampered with, or stolen.

When it comes to data, why should we worry about security?

Businesses rely on the data center for storage and access to critical information, applications, and services; as such, it is imperative that the data center be well protected.

Lack of proper data center security can lead to the exposure or theft of valuable corporate or, even worse, consumer information.

A data breach of this nature may have severe repercussions for a business, both monetarily and in terms of public trust.

The requirement for infrastructure-level data center security is growing as more data center technologies become virtualized.

Safety from harm

Data centers of different sizes require different levels of physical security.

There is a wide variety of information technology (IT) hardware in data centers, including servers, switches and routers, power and cooling infrastructures, and telecommunications gear.

In a closet, where a simple lock provides adequate security, or a warehouse, this gear may be stored.

Badge access, video monitoring, alarms, and armed guards are all examples of physical security measures that might be supplemented with more precautions.

Fire safety is another important aspect of building protection.

Chemical fire suppression systems, rather than sprinklers, are recommended for protecting the electronic equipment found in a data center from the risk of fire.

Safety in cyberspace

With virtualization, the servers, networks, and storage in a data center may all be treated as separate entities.

As a result of this abstraction, IT managers may control data center services from afar, automating routine tasks using software and distributing workloads over several servers in real time.

When it comes to virtual security technologies like firewalls and intrusion prevention and detection systems, some data center networking software already offers security as part of the offering.

This program might be used by IT administrators to establish rules for user authentication and access privileges in the data center.

Information stored in a data center may be backed up using the same software that prevents unauthorized individuals from accessing or stealing that data.

Protecting Corporate Networks

While the network is increasingly important to the success of your digital company, it also presents new security risks.

The complex nature of modern threats, such as credential theft and encrypted assaults, makes it challenging to secure the ever-expanding perimeter.

Malicious cyber security threats are becoming both more common and more complex.

Organizational cyber defense must include measures to strengthen the security of internal networks.

Security measures including virtual private networks (VPNs), remote access management software, intrusion detection systems, and wireless network encryption are all safeguarded by a hardened network.

The security offered by intent-based networking is cutting edge.

Advantages and drawbacks

- Consolidate security measures.
- Protect the network with uniform policies that apply to all users, all applications, and all locations.
- Limit exposure to dangers
- Common exploits may be prevented with the use of automated security patching, DNS protection, and wireless security.
- Gain complete awareness.
- Discover the network's users, their gadgets, and their encrypted traffic.
- Deployment Streamlining.
- Cisco DNA Center settings may be used to rapidly roll out new security additions.

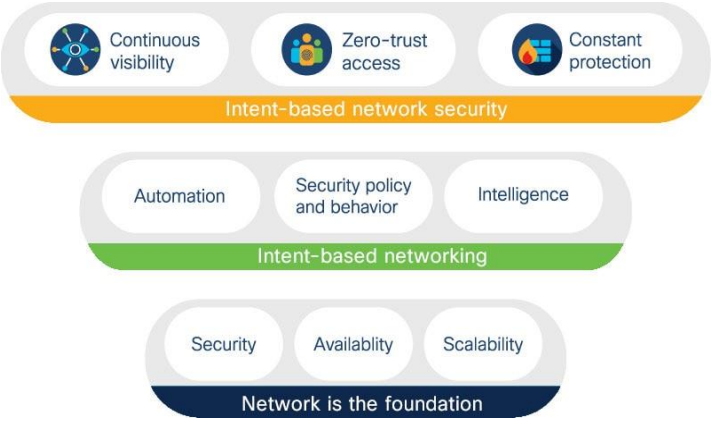

Figure 1.2: Enterprise Network Security Architecture. (Copy right is reserved to William Stallings.)

1.4.2 Protecting a WLAN

Challenges

WLANs have allowed workers greater independence and increased productivity inside and outside the office.

Even as productivity has soared, though, new security threats have emerged.

Wireless signals are intended to travel outside a company's physical limits, putting into question the long-held belief that internal networks are safe.

Unauthorized users, or even hackers, may be able to intercept and use signals coming from open WLANs that go beyond the company's internal network.

Its transient nature necessitates a robust security architecture.

Risks to the network can be reduced in five stages.

wireless dangers

- Construct a wireless LAN security plan.
- Keep the wireless network safe.
- Protect the Ethernet network from intruders using wireless technology.
- Protect the company against any potential dangers.
- Involve your staff in keeping the network safe.

1.4.3 Wireless LAN Security

Guests once only had access to WLANs in conference rooms, but now there are "hot" zones of connectivity all across an organization, and eventually there will be complete coverage everywhere.

To secure a WLAN requires little effort. The Cisco Unified Wireless Network and other technological breakthroughs in the sector have made this simpler than ever before.

Expanding on the three tenets of the Self-Defending Network concept, network security consists of:

Safe and sound transmissions

Policy for managing and containing threats

Management of compliance.

Safeguard the Ethernet network from wireless intrusion.

Data encryption and user authentication are essential components of secure communications.

Like a wired network, a wireless one does not require a combination of these two components, though doing so is strongly advised.

A few such exemptions include guest or hotspot networks.

Moreover, the wireless medium's peculiarities necessitate the use of supplementary security measures to protect the network.

1.5 Lan Switch Security, Router and Switching Mechanisms For Security.

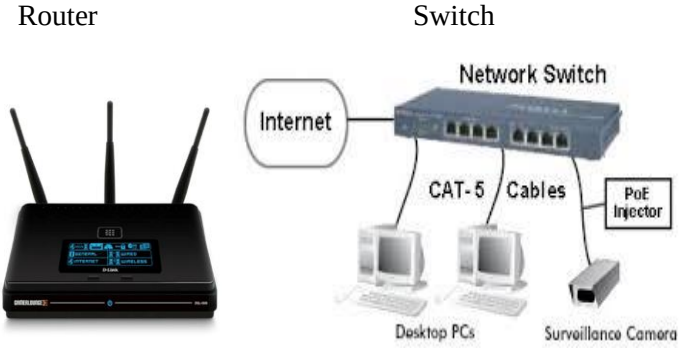

Figure 1.3: Router and Switch (Copy right is reserved to William Stallings.)

1.5.1 Switches

Hubs and switches in a network connect different parts of a network that share a physical media.

When data is sent across one section of a network, it is relayed to the other sections via the hub.

Because all they do is accept and relay information, hubs are labeled as "dumb" devices in a network.

While switches are the "smart" network nodes.

Each ARP reply is mapped to a specific switch port, and the switch keeps track of this information in its own internal ARP database.

When a package arrives,

The switch learns the MAC address of the destination and

Instead of broadcasting the packet, it is directed to the specified port.

When the destination MAC address is unknown, switches only retransmit the message to all ports.

Resolving Protocol for Addresses

It exemplifies a type of protocol for gaining access to a service.

According to RFC826, it helps the network layer (layer 3 of the OSI model) by translating IP addresses into hardware identifiers.

In order to convert a hardware address to an IP address, the Reverse Address Resolution Protocol (RARP) is used.

Function codes like are included in ARP packets.

Initiating an Address Resolution Protocol

Reply to ARP,

Inquiry for RARP

RARP response.

The query data is included in a broadcast packet that is sent in response to an ARP or RARP request.

1.5.2 Security Routing and Switching Systems

In the OSI paradigm, connecting with other networks is only possible through the use of routers.

Network routing can be blocked even though the physical and data connection layers are still operational.

There are several types of router-based attacks:

Direct assaults on the router cause the least amount of congestion on other networks.

A Poisonous Meal,

Inundating Tables,

The use of Metric Attacks

o Looping attacks against routers.

Routing Protocol Exploits

• A denial-of-service (DoS) or system breach is the hallmark of a direct attack on a router.

• A denial-of-service attack can prevent a router from functioning, breaking the connection to the internet.

• These attacks are load based, meaning that the router will fail to handle traffic if the volume of data passing through a certain network interface is too large.

Although most personal computers have powerful processors, hardware routers often employ much slower CPUs. While hardware-based fixes usually work quicker than software-based ones, weaker CPUs require less work to crash.

Example

It's possible that a computer with 2 GHz of processing power with Linux installed may serve as a stable router.

Whereas a Cisco PIX, a commercial router with a 60 MHz CPU, is widely deployed.

• Router-Dependent: The router's settings can be changed to forward the data to a new server,

blocking traffic from particular hosts or allocating new,

Secret tiers of the internet.

• A compromised router can undermine the security and privacy of any networks it is linked to since it spans numerous subnets.

Poisoning a Router Table

1.ARP table-related

- The routing table at the network layer is susceptible to poisoning attacks, just as the address resolution protocol (ARP) table in the data link layer.
- Only a small fraction of network protocols really verify packet authenticity.
- Compromised or spoofed network traffic
- □ overwrite,
- or insert
- Take out the items in the routing table.
- The end consequence is not dissimilar to that of a vulnerable router.
- Rearranging the contaminated table
- forward the data to a new server,
- blocking traffic from particular hosts or allocating new,
- Secret tiers of the internet.

2.As a result of changes to routing tables,

- Dynamic routing tables are supported by network layer protocols; these tables are produced and modified in real-time based on network activity.
- Why are these protocols more susceptible to attack?
- First, data on poisoning may be produced by a brand-new node.
- Two, there aren't a lot of verified dynamic nodes.
- Static routing tables should be used on critical routers to prevent poisoning.

Disaster on the Router Table

Attack on memory:

When it comes to memory and storage space, routers typically fall short.

There is often a cap on the size of the routing table.

Devices that don't rely on static routes have the added responsibility of handling route expiration and table filling.

An adversary can fabricate information that the router will use to construct its routing table.

When the router's routing tables are full, it can do one of three things:

– ignore,

– Remove the oldest

- exclude the poorest possible paths

When the router's routing tables are full, it can do one of three things:

– ignore,

– Remove the oldest

- exclude the poorest possible paths

1. Ignore Alternative Paths:

Ignoring new routes is an option for routers.

Established table entries are safe from an attacker, but they can block genuine new entries from being added.

2. Get Rid of Tired Paths:

Routing tables, like ARP tables, may drop unused paths.

Intense flooding can displace long-standing entries in a table.

3. Get Rid of Inefficient Paths:

Routing table metrics may be taken into account by the router. Alternatives to less favorable routes can be substituted.

While new routes that look desired are not likely to displace default or highly preferred routes, they can dislodge table entries for other routes.

4. The success of a router table flooding attack depends on the attacker's familiarity with the router's behavior when its routing table is saturated.

5. An assault is successful, for instance, if it creates a large number of seemingly optimum but really phony pathways.

If the router drops the lowest-quality paths,

However, the attack will be mitigated if the router does not take into account any new pathways.

6. Static entries may usually be added to a dynamic routing table.

7. Important routes should be set up as permanent entries in the routing table.

Loophole Exploits in Routers

• Network protocols often make an effort to discover and stop network loops.

• Data sent through one interface of a network loop is sent through another interface of the same router. When there is a

network loop, all of the available bandwidth is used, which is a major problem.

• Many methods exist to aid in the discovery of network loops, while certain network layer protocols give ways to expire undelivered messages. Systems protocols like

Spanning Tree (IEEE 802.1) in the data connection layer, and

Network layer Border Gateway Protocol (BGP) [RFC1772] and

[RFC2453] The Routing Information Protocol (RIP).

When a network router detects a network loop, it removes the path from the routing table. A looping attack causes the router to falsely identify a network loop by generating a fake reply to a loop check.

• A result is the deactivation of a useful route in the network. Table and metric assaults can be prevented by using static values, although pathways can still be disabled by using a looping attack.

The good news is that looping assaults are notoriously hard to pull off.

Each router interface an attacker targets requires two separate systems under their control.

• As soon as one system detects the loop-check inquiry, a message with the response is transmitted to the other.

1.5.3 Switch Security

MAC address identification

There are three methods in which a switch can determine a MAC address:

When the switch encounters an unknown destination, it sends out an ARP request on all of its ports and waits for a response.

The ARP database of a switch can be populated and updated by any ARP reply, regardless of where it originated.

Each message frame has a source MAC address in its frame header. This allows the switch to assign a port to the source MAC address.

Switch Attacks

Switches and bridges are susceptible to poisoning and flooding attacks in the same way that any other network component is.

Switch Poisoning Attacks

In order to direct data packets to their proper physical network ports, switches keep an ARP table. The ARP table can be tainted by a poisoning attack.

The MAC address of another node can be assigned to a different port via a poisoned ARP reply.

All communication destined for the target node is redirected to the attacker's port, essentially isolating it from the network.

To hijack a connection, an attacker must first divert network traffic through a "poisoned" switch.

Switch Flooding Attacks

A switch's ARP table can be flooded by an attacker employing ARP poisoning.

When the ARP database of a switch becomes full, the switch typically returns to the hub mode since it cannot afford to discard packets.

In this configuration, all nodes are active and receiving data.

In promiscuous mode, a node can start receiving all data sent across the network.

Static ARP entries are supported by most bridges and high-end switches.

Switch flooding and poisoning can be reduced by employing static ARP tables in these network gadgets.

1.6 Queuing & Scheduling Algorithms

1.6.1 Queueing And Scheduling

By applying queuing and scheduling to an interface, traffic may be segmented into many queues, and the scheduler can then determine the processing policies for each queue individually.

As shown in the accompanying Figure, the scheduler can implement differentiated behavior for several classes of service if the traffic mapped to each queue is of a specified class.

After a brief review of the material covered thus far, let's go further into the inner workings of queuing and scheduling.

Buffering, or the length of the queue, refers to the amount of memory available to temporarily hold packets, whereas bandwidth refers to the rate at which packets may be processed.

However, as we shall see, it is not always necessary to queue the complete packet; often only a notice, which is a representation of the packet contents, is maintained.

If queuing is enabled on an interface, the buffering value can be set as the amount of time during which packets are allowed, or as the physical size of the queue in terms of the maximum number of packets or packet notifications.

The buffering value, either in milliseconds of traffic or absolute, is a limit on the amount of memory that may be used.

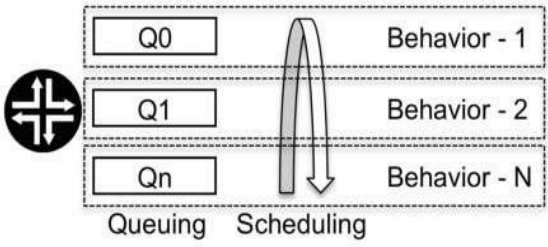

Figure 1.4: Using a variety of strategies for queuing and scheduling. (Copy right is reserved to William Stallings.)

Queueing And Schedule Width Parameter

A total amount of bandwidth is made available to the queuing and scheduling mechanism, and scheduling determines how much of that bandwidth is allocated to each queue. This total amount of bandwidth can be the interface speed, or the shaping rate if a shaper is applied after the scheduler, as shown in figure.

If there are enough resources and no competition for them, then there is no need for queuing. Congestion can be caused by sending more data through an interface than the outgoing line can handle.

By applying a shaping rate to the interface that imposes a speed limit below the maximum interface line speed, artificial congestion can be created; the remaining traffic is then throttled or back-pressured into memory and partitioned across the actual queues, with transmission rates for each queue determined by the scheduler.

In a queue, a packet joins at the end and stays until it reaches the front, at which point it is removed.

It is possible to remove packets from the queue at either the tail or the head, or from both locations simultaneously; however, this is rarely done.

When the queue buffer fills up at a rate greater than the removal rate, all incoming packets must be deleted since there is no more room in the buffer.

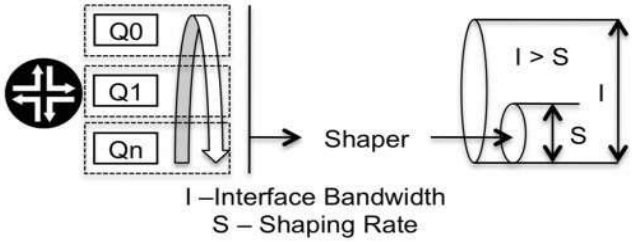

Figure 1.5: Bandwidth as a Scheduling and Queuing Parameter

(Copy right is reserved to William Stallings.)

Losses in tail and head age occur in a queue

It is possible to remove packets from the front of a line, which means that the packets at the front of the queue are the ones that have waited the longest.

A maximum time is enforced on all packets in a queue for how long they are allowed to remain in the queue, waiting to be scheduled, to prevent the queue from stalling and having a hopelessly long delay for the traffic inside the queue.

This phenomenon is known as packet aging, and it describes the process by which unwanted packets are removed from a delivery queue buffer.

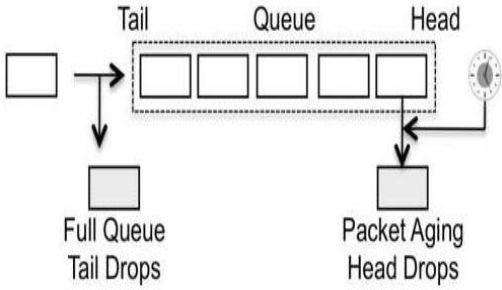

Figure 1.6 Linear decline in head and tail age. (Copy right is reserved to William Stallings.)

Embedding cells into packets

The older and conceptually simpler method for handling packet queuing in routers is to queue the entire packet, as is done in most CPU-processing-based forwarding routers. However, this approach has the disadvantage of not separating the control and forwarding planes, which can be a limiting factor if the number of supported interfaces is low.

However, cellification has emerged as a preferred alternative due to the disadvantages of queuing the complete packet. A packet has a variable length, with minimum and maximum

values determined by the technology or, optionally, by the interface configuration.

Different packet sizes require different amounts of time to process because each packet has to be read byte by byte each time it is processed by the router CPU. Additionally, a packet can contain several different headers, which adds to the difficulty of managing memory resources.

By dividing a packet into cells of the same size, the cellification process simplifies the memory management required by the router.

Consistency in transmission timings and buffering time slots is also provided.

A packet with several headers is seen being "cellified" in the adjacent Figure; these cells have a total size of 64 bytes, and one of these cells is designated as the "notification" or "cookie" for the purpose of further communication.

The cookie stores the packet's essential data for further processing by the router. So, if the router has to analyze a packet, it only processes the cookie and not the complete packet.

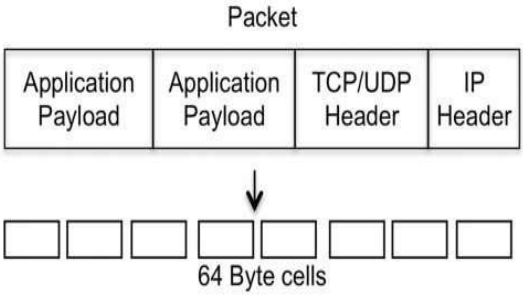

Figure 1.7: A Packed With Cells (Copy right is reserved to William Stallings.)

The goal is to build a model of the packet that contains just the necessary data. For instance, the cookie stores the DSCP value of X if the IP packet's header includes that value. However, the cookie cannot contain any additional data, such as the packet payload.

The information included in a cookie might change based on the services to which a packet has been sent. In addition, the cookie is updated in response to any changes in the packet header, including the DSCP field.

In most cases, the router's setup and management are unaffected by the cellification process. Nonetheless, you should familiarize yourself with the fundamentals and the advantages it provides.

1.6.2 Types of Queueing Algorithms

Over the course of many years, several distinct queuing algorithms have been developed as a result of a number of mathematical investigations. All potential techniques for queuing would be difficult to define. However, the following parts of this article cover some of the most well-known scheduling disciplines:

FIFO Queuing System (First-In, First-Out)

FQ - Fair Queuing

The PQ System (Priority Queuing)

(WFQ) Weighted Fair Queuing

WRR, or weighted round robin

Shortfall-based round-robin (DWRR)

Deficit-weighted round-robin with priorities (PB-DWRR)

PB-DWRR is the most potent of them and maybe the most intriguing as well. However, this article covers all of them, since comparing and contrasting their relative merits and shortcomings is fascinating and as doing so sheds light on the development of queuing algorithms and the reasons behind PB-DWRR's widespread use.

1.6.3 FIFO Queuing Algorithm (First-In, First-Out)

First-in, first-out, or FIFO for short, is a widely used queue scheduling principle. FIFO queuing works on the idea that all packets should be placed in the same queue and processed in the

order in which they were received. Thus, in brief, the scheduler only works with a single queue.

If a shaper is used, this method suggests that the shaping rate, rather than the interface speed, determines the removal rate. Because no packet can pass another within a queue, packets are processed in the order they were added to the queue.

FIFO is likely the default behavior for the vast majority of router manufacturers, with a separate queue for locally generated control plane packets like routing protocol packets and another for most transit data.

In order to construct packets before sending them, the vast majority of hardware implementations require at least one buffer per interface. The queue serves as a temporary storage area for packets before their Layer 1 and Layer 2 headers are appended. In order for the system to make effective use of the interface at line rate, it benefits from having a buffer of the size of two to four packets.

With FIFO service, you can count on consistent performance, and the algorithm is easy for a router or switch to implement. Since vehicles from various service classes all utilize the same road lane (essentially the same queue) to the scheduler, the FIFO method may not be a genuine "queuing and scheduling" solution.

When the queue fills up, all newly arriving packets are deleted, hence FIFO queuing causes a delay under congestion

situations. Here is an area where drop behavior implementations can vary.

To prevent the queue from collapsing due to blocked sessions, one option is to discard everything in the buffer. This action is taken so that the receiving window doesn't grow too tiny when packets in the buffer memory pile up and wait to be processed, as shown in the TCP silly syndrome.

As a result, all sessions experience a decline in quality of service, and packets remain in the queue in the order in which they were received. Large amounts of buffer memory may be available, but they are underutilized since only a fraction of the cwnd available to senders is really being used.

Dropping from the tail of the queue and employing packet aging are two more sophisticated options for clearing the backlog. The advantages of first-in, first-out (FIFO) queuing are as follows:

• It is easy to implement the FIFO algorithm.

Removal from the queue is optimal for delay-tolerant TCP applications; however, this comes at the expense of packet ordering being maintained. When moderate congestion arises, TCP slows down due to RTT, but retransmissions are kept to a minimum.

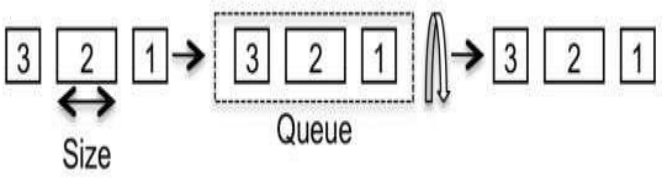

Figure1.8: FIFO Scheduling (Copy right is reserved to William Stallings.)

Among the disadvantages of FIFO queueing are the following:

• There is no special treatment for FIFO customers. When there is a lot of traffic and the buffers are full, all services are almost useless.

• Since the queue depth is dynamic, delays and jitter cannot be mitigated. As a result, FIFO is not a viable option for use in time-sensitive programs. A TCP transfer of 1500 bytes with a big cwnd, for instance, might block a voice flow consisting of numerous packets.

• Bursty flows can exhaust all available buffer space if greedy flows occupy the majority of the queue depth. Session control is difficult with a single queue or buffer because TCP always seeks to increase the size of the transmitting window.

1.6.4 Fair Queuing

Since FIFO does not distinguish between flows or streams of packets, the fundamental drawback of FIFO queuing is that

flows consisting of numerous packets might eat up much of the bandwidth for less bandwidth-intensive applications. The primary drawback of first-in-first-out (FIFO) queuing is overcome by the scheduling algorithm known as fair queuing (FQ), also known as the fairness algorithm.

The fair scheduling technique provided by FQ is made possible by the classification of packet flows into different queues. By segmenting traffic and flows, FQ prevents light-weight applications from being starved by heavy-weight ones.

With queues buffering packets that belong to various flows, scheduling is mostly a statistical multiplexing procedure across queues.

Nagle's definition of FQ from 1985 accounts for packet size to provide each flow a fair shot at sending the same amount of data. Equal bandwidth is allotted for each queue, meaning that each queue has the same priority.

Since the pace at which packets are removed from the queue is a function of the quantity of bits, rather than the number of packets, this technique circumvents the difficulties associated with handling both tiny and big packets in the queue.

Therefore, in each scheduling turn, the queue is serviced in terms of number of bits, not number of packets, meaning that a queue with large packets has access to the same bandwidth as a queue with little packets.

Providing an unequal allocation of resources across different classes of service is a primary purpose of the quality-of-service

(QOS) framework, therefore the fact that the fairness algorithm is, well, fair, is a major drawback. Flows that need minimal latency may be impacted if several queues must be accessed in a predetermined order.

In order for routers to follow the FQ paradigm, a hash function is required to classify flows independently so that packets may be mapped to a specific session. The session or flow may be computed by the hash function using a wide variety of parameters, including the source and destination ports, the protocols used, and perhaps even information at a higher layer than TCP, UDP, and port numbers.

Dynamically allocating memory and creating logical queues that exist only when the flow is active is a very resource-intensive task that is difficult to implement when most of the packet forwarding takes place within a hardware-based architecture that has little interaction with processing resources. Therefore, it is not scalable to categorize and dynamically create queues for each running hashing flow.

Oversubscribed backplanes on routers and switches are often dealt with using the fairness algorithm. A one-stage or two-stage fabric switch connecting line modules is a popular hardware layout.

In this setup, data is sent from several source ports via a fabric to a single receiver port. fairness can be used to guarantee that all incoming ports on line modules have an equal chance of

being delivered to one of the outgoing module's ports. However, the fairness algorithm does not account for the fact that the backplane is unaware of individual flows, which is the inspiration for using FQ to achieve bandwidth fairness. Therefore, a variant of the original FQ algorithm is in fact being used.

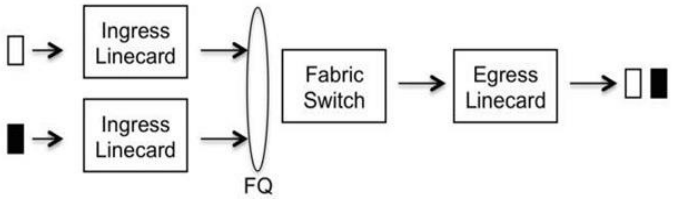

Figure1.9: Fairness Algorithm (Copy right is reserved to William Stallings.)

Since packets are distributed sequentially to guarantee fairness in each scheduling cycle, the FQ method indicates that the data rate may vary for brief durations. This tendency may reduce performance, especially in comparison to algorithms that prioritize throughput maximization. On the plus side, FQ is excellent at preventing flow starvation under large loads. The following is an advantage of FQ:

Each incoming stream is placed in its own virtual queue thanks to the FQ algorithm. As a result, a greedy flow should have no practical effect on other queues.

FAQ has following limitations

• High-speed routers or routers designed to handle a large number of sessions and a high volume of traffic have yet to see a

vendor implementation of the FQ algorithm because of its complexity. FQ is more of a conceptual framework than a working paradigm.

• It uses a lot of resources because of the large number of states and hashes calculated and the need to constantly reallocate memory in response to session state changes.

• Each session hash is treated as a queue object, thus delay and jitter may still be problems. If there are a large number of sessions that need to be scheduled, the session slot interval can be determined using a multiplexing system that takes into account the number of sessions that are now sending packets. If there are several open sessions, the period before the next service is provided may be considerable.

1.6.5 Priority Queuing

When compared to other, more sophisticated queuing disciplines that also offer differentiated service, the PQ requirements for queuing and scheduling are low and not particularly complex.

But PQ has its own restrictions. It's possible that lower priority queues will run out of resources entirely if the volume of high-priority traffic increases to an unsustainable level. Dropped packets might originate from the queue's tail or its head, or both,

if the queueing rate remains constant while the removal rate declines.

As the buffer space for the low-priority queues begins to overflow, the proportion of dropped packets from that queue rises. As a result, not only are packets being lost, but latency is also increasing. In the case of Transmission Control Protocol (TCP) sessions, retransmission of traffic might cause its state to become outdated.

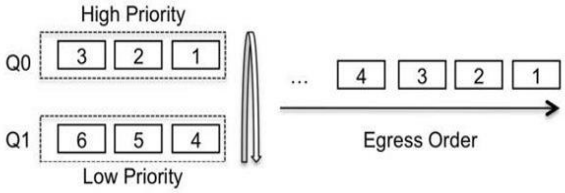

Figure 1.10 PQ scheduling. (Copy right is reserved to William Stallings.)

The advantages of PQ are as follows:

The PQ algorithm offers a straightforward approach to accommodating diverse service classes, in which queues are treated differently based on their given priorities.

Implementing PQ on high-speed link network equipment has modest processing needs for buffering and scheduling.

Low-delay and loss-sensitive applications, like real-time traffic, may be successfully shielded from other greedy traffic

flows, like TCP sessions, while high-priority traffic can keep its delay and jitter to a minimum.

The following are some of PQ's drawbacks:

Network failure can occur when low-priority traffic is halted due to the PQ algorithm's strict allocation of resources and scheduling services to priority-classified traffic. When some traffic is given precedence over others, other traffic must be sacrificed.

It is crucial to properly offer and regulate the rates for high-priority traffic. If not, low-priority connections might experience interruptions in service.

1.6.6 Weighted Fair Queuing

For this reason, weighted fair queuing (WFQ) is also referred to as "bit-by-bit round robin," or simply "bit-by-bit," since it implements a queuing and scheduling system in which queue servicing is based on bits rather than packets. In an effort to simulate the virtual processing time afforded by Generic Processor Sharing (GPS), WFQ was created by Demers, Keshav, Shenke, and Zhang back in 1989.

In WFQ, the interface rate or the shaping rate is used to calculate a weight for each queue or flow. WFQ is sensitive to packet sizes and can accommodate packets of varying sizes. Since WFQ effectively prioritizes bits over packets, it ensures

that sessions with large packets are not given preferential scheduling treatment over sessions with smaller packets. Therefore, sessions with lesser packet sizes will not be treated unfairly when scheduling. WFQ uses a computation on the packet's bits at the queue head to determine when each packet will be processed.

WFQ inherently increases complexity since traffic processing is performed based on a stream of bits rather than packets, and because what the router actually receives and sends are packets.

Since each queue in Figure 1.10 has the same weight, the amount of bytes planned in each scheduling turn also reflects the weight value. Three packets worth X bytes are removed from queue 0 (Q0), one packet worth Y bytes is removed from queue 1 (Q1), and two packets worth Z bytes are removed from queue 2 (Q2). The weighting factor serves as a cap on the total amount of resources that may be allocated and utilized.

The bit calculations required by WFQ are exceedingly resource-intensive, which is a major downside of the algorithm. Due to the lack of aggregated flows into classes with finite queues, the original WFQ concept is also quite resource intensive. Instead, similar to FQ, a separate queue or buffer quota is allocated to each flow or stream.

WFQ has been deployed more on CPU-based platforms whose queuing disciplines are based on bus-based architectures

due to the high resource requirements and complicated computation required to monitor the state for each flow and its packets.

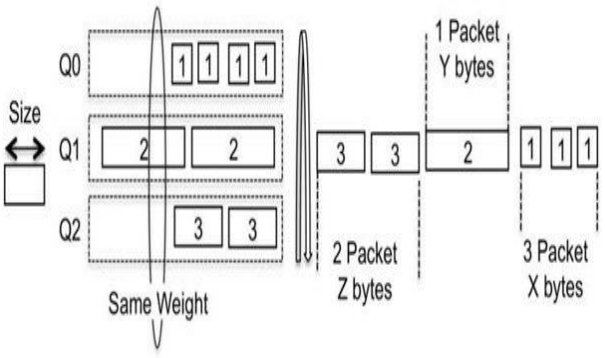

Figure 1.11 WFQ scheduling (Copy right is reserved to William Stallings.)

The advantages of WFQ are as follows:

Implementations based on the WFQ algorithm distinguish between not just individual flows, but also between classes and their aggregated traffic. The scheduling and bandwidth ratio is divided according to a weight assigned to each class. WFQ is also flexible in terms of packet size, as it takes into account individual bits.

Some restrictions of WFQ are as follows:

The first iteration of WFQ was conceived as a queuing theory. Current implementations deviate from the envisioned solution in which a weight is assigned to each flow. Instead,

flows are aggregated by being sorted into several service classes and then allocated to respective waiting lines.

The implementation of WFQ is just as difficult as that of FQ. Both calculating the hash table and keeping track of state information need a lot of processing power.

1.6.7 Weighted Round Robin

When it comes to scheduling, weighted round robin (WRR) is a discipline that can make up for what PQ and FQ lack. WRR's fundamental idea is to schedule classes with various bandwidth needs.

WRR is able to do this because it permits a number of packets to be withdrawn from a queue whenever that queue is scheduled. WRR also solves the problem that PQ has when a high-priority queue might starve lower-priority queues. WRR achieves this by permitting the deletion of at least one packet from each queue during each scheduling round. The similarities between WRR and WFQ are striking at first look.

As for the distinction between WFQ and WRR, the former services bits in each scheduling turn while the latter processes packets. The packets that will be processed during a given scheduling round is determined by the queue's density.

The service disparities across queues and traffic classes are reflected in the weight, which is often expressed as a percentage of the interface's bandwidth.

It depicts the elimination of three packets from queue 0 (Q0), as well as a single packet from each of queues 1 and 2. The importance of a packet in the queue may be gauged by the rate at which it is deleted. Since Q0 is assigned three times as much importance as Q1 and Q2, it also eliminates three times as many packets on each scheduling round.

To schedule packets, WRR does not know their actual sizes in the buffers. In most cases, the queues and scheduling will work well for a medium-sized packet. The sizes, however, are only estimations with no bearing on the actual distribution of traffic in each queue.

This WRR function has benefits and drawbacks. Unlike WFQ, which must convert bits to bandwidth scheduling, WRR requires no sophisticated resources that necessitate state computation, making it easy to implement.

As a result, WRR becomes something of a core QOS solution that can handle both high traffic volumes and congestion, making it ideal for managing a large number of flows and sessions.

The problem with WRR is that it cannot deal with packets of varying sizes, hence it is blind to the available bandwidth. A packet of 1500 bytes is the same as a packet of 64 bytes in terms of getting a scheduling turn.

In reality, packet weight is only taken into account once each service scheduling cycle. Over time, WRR can produce acceptable fairness in scheduling when the traffic mix is relatively uniform across classes and queues. However, there may be substantial variations in the immediate future. In addition, if there are significant changes in packet size between traffic classes due to variances in traffic mix or traffic type, the queueing situation might become very unbalanced in favor of the classes that are dominated by big packets. As shown in Figure 7.10, a traffic class using TCP can benefit from the larger packets of a non-real-time traffic class.

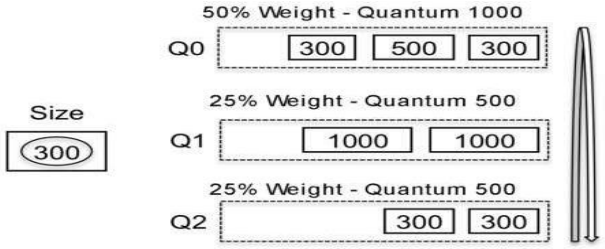

Figure 1.12 Weighted Round Robin scheduling (Copy right is reserved to William Stallings.)

Figure's queues Q1 and Q2 are equivalent in importance. However, Q1's bandwidth rate is in fact double that of Q2's since Q1's packets are twice as large as Q2's.

The advantages of WRR are as follows:

The WRR algorithm can be implemented with little effort.

The following are some of WRR's drawbacks:

• When queues include packets of varying sizes, scheduling might be unfair due to WRR's inability to account for them. When one queue is filled with little packets and another is filled with large ones, more bandwidth is given to the latter.

Services with stringent latency and jitter requirements may be adversely affected by changes in the scheduling priorities of other queues. There are no different scheduling priorities available on WRR.

1.6.8 Deficit Weighted Round Robin

When using WRR, the amount of packets that are serviced during a given scheduling turn is determined by a weight that is proportional to the bandwidth available to that queue. As was said before, when average packet sizes vary between queues and flows, bandwidth allocation can become unjust.

For queues with typically smaller packet sizes, this behavior might lead to a decrease in service. To overcome the shortcomings of Weighted Round Robin (WRR), a new scheduling discipline called Deficit Round Robin (DRR) or Deficit Weighted Round Robin (DWRR) has been developed. M. came up with the first concept. Two of our staff members, Shreedhar and G. In 1995, Varghese was involved. Deficit algorithms can process packets of varying sizes without needing to know their average.

Packets that are longer than a certain threshold are queued until their turn comes up in the scheduling process.

Packets at the front of every non-empty queue with a deficit counter larger than the size of the packet at the front of the queue are served by DWRR, whereas WRR serves all non-empty queues. The queue is bypassed and the credit value, or quantum, is increased based on whether or not the deficit counter is lower. The next time the scheduler checks the waiting list, it will use the new number to determine the deficit counter. When a packet is delivered from a queue, the credit is reduced in proportion to the packet's size.

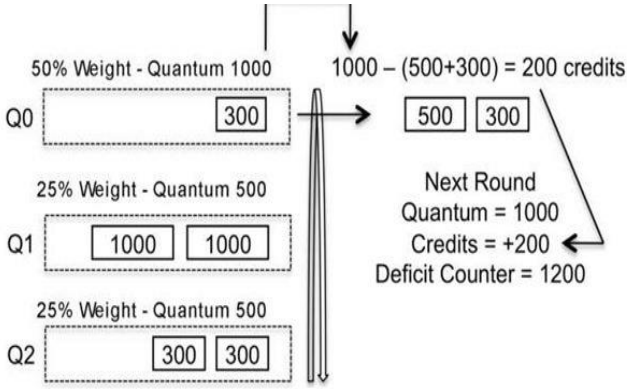

Figure1.12 DWRR scheduling, turn 1, Q0 (Copy right is reserved to William Stallings.)

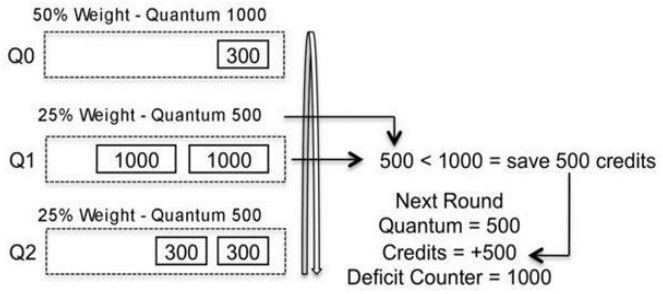

Figure1.13 DWRR scheduling, turn 1, Q1 (Copy right is reserved to William Stallings.)

The implementation of DWRR depends on the following elements and factors, all of which influence the scheduling service for the queues:

Weight, like the weight in the WRR algorithm, is a representation of the bandwidth available on the interface that is sending data.

The quantum reduces the mass value to a string of bits. Each scheduling round results in an increase in quantum value. Therefore, the quantum is the bandwidth-aware scheduling technique that acts as a throttle.

Credits can have a positive or negative monetary worth. When a scheduler turn ends with unused bytes remaining in the queue, positive credits are awarded. This number will be used at the next available scheduling time in the queue. When a queue has sent more data than its bandwidth value allows in a given scheduling turn, it will end up with a negative credit balance.

When added together, the quantum and credits form the deficit counter, which ensures bandwidth equity. Once the deficit counter approaches 0 or the packet size exceeds the remaining deficits, the scheduler begins removing packets. If there aren't enough deficiencies in a queue to schedule a packet, the scheduler will skip that queue and continue on to the next one.

Let's use some real-world examples of DWRR to highlight its features. where three queues have various weights that represent the relative amount of bandwidth available on each outgoing interface. The first queue (Q0) receives 50% of the bandwidth, the second queue (Q1) receives 25%, and so on.

To keep things simple, let's assume that the length of each queue is the same. Each queue stores a varied number of packets of varying sizes. Each queue's quantum number is credited with a certain amount of bandwidth during each scheduling round. Scheduling PQs.

The dequeuing procedure and the first scheduling round for Q0 are depicted in the figure. Two of Q0's packets are 300 bytes in size, while the third is 500 bytes in length (see Figure). Quantum value Q0 is increased by a factor of 1000 every scheduling round. Since the deficit counter is currently at 1000, we can remove 1000 bytes from the queue at this turn. Since packets of 300 and 500 bytes each eat up 800 of the available 1000 credits, they can be sent. However, the final 300-byte packet is too large to send in this round (1000 minus 800 = 200)

due to a lack of available credits. As a consequence, 200 credits from Q0 are held over until the next scheduling turn, and the process advances to the next queue containing packets in order. Here we will examine Q1, the next queue in line.

Q1 has two packets of 1000 bytes each, however the quantum is 500 credits every scheduling round. There aren't enough credits to send any packets during this scheduling cycle (500 1000). As a result, 500 more credits will be added to the Q1 deficit counter before the next scheduling round. The power of deficit-style algorithms over standard round-robin systems is demonstrated here. Due to the fact that Q1's packets are less in size than the allowable quantum and credit values, Q1 is penalized by DWRR. This demonstrates that DWRR scheduling takes into account both the total number of packets and their varying sizes. Let's go to Q2.

Scheduling for the DWRR, round 1.

The quantum value of the packet at the front of Queue 2 is 500, and since it is only 300 bytes in size, the packet gets sent. Subtracting 300 from 500 yields 200 credits, which are put aside for the next scheduling round. the scheduling wheel has rotated once and has landed back at Q0.

It is currently 1200 on the Q0 deficit tally. After sending the 300-byte packet, a deficit counter subtraction of its value yields a new deficit credit value of 900. Depending on the specific

implementation, the deficit counter may or may not be reset to zero if there are no packets waiting in the queue. The algorithm respects such queues, but gives them no credit for doing nothing. In order to achieve fame and glory, the queue must be completely full. The scheduler is now focusing on the problematic Q1, which failed to send any packets during the first scheduling round.

Positive news for the first quarter has arrived at last. With the current quantum value of 500 and the extra 500 credits from cycle 1, Q1 has enough to send one of the 1000-byte packets. The outlook for the second quarter is bright. The final packet is sent since the deficit counter is much larger than the number of bytes in the queue.

1.6.9 Priority-Based Deficit Weighted Round Robin

The WRR algorithm gains awareness of bandwidth and enhanced fairness thanks to the introduction of the deficit counter. However, fairness is not the intended action in some forms of traffic. What's needed is a PQ-style priority scheduling that keeps the advantages of DWRR intact. Predictable service for critical, real-time traffic requires the introduction of a priority level for scheduling.

Service assurance in terms of delay and loss prevention may be provided for demanding traffic types like speech and real-time broadcasting by enabling stringent priority or by providing

multiple priority levels and employing DWRR scheduling across queues with the same priority levels.

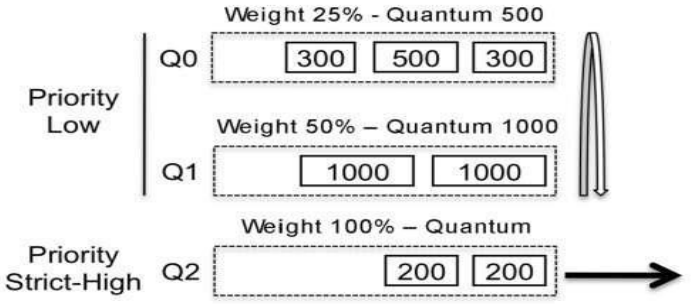

Figure1.14 PB – DWRR scheduling, with one strict – high priority queue (Copy right is reserved to William Stallings.)

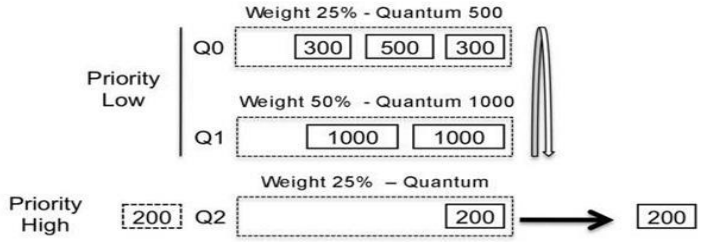

Figure1.15 PB – DWRR scheduling, policed priority – high queue (Copy right is reserved to William Stallings.)

Scheduled service cycles must use the more stringent Q2 queue. priority queue, which always takes precedence over the other two. Q2 is depleted first in every scheduling round. Since Q0 and Q1 are not functioning according to their stated weights, packets in them get stale and can age out if the strict-high queue fills up faster than it is cleared.

The fact that a strict-high queue never enters negative credits has unintended consequences, including the aging out of packets. But a rigorous. If the pace of packets entering Q2 is managed, for instance by limiting them to 20% of the total bandwidth, a high priority queue system can function. This is reflected in the credit status. Adding a policer that drops packets beyond a preset rate limit is one technique to regulate the flow into a queue. The strict-priority queue will not go into a runaway condition where it consumes all available scheduling cycles if this plan is implemented.

Managing the queue's waiting time is another option for limiting new arrivals. Since Q2 can never have more than one packet in it due to its scheduling weight of 25%, Q2 must discard the 200-byte packet at the end of the queue.

As an alternative to a strict-high priority queue, an alternating priority scheme may be used to set limits on the queue's throughput. Using this strategy, the queue with the highest priority may be slotted in between the other queues. For instance:

However, because other queues may impose delay based on the traffic that they plan, alternating priority mode may result in some unanticipated delay. Furthermore, the pace of the queue cannot be managed while using alternating priority. Queues can also be made to act in accordance with their weight, with priority being ignored if this weight is exceeded.

This method avoids the problem when packets in lower-priority queues get outdated due to higher-priority queues running amok. In this example, there are four queues of varying priority. Due to the varying importance of tasks, hierarchical scheduling is necessary. Priority is given to Q2 and Q3 in the queuing system. These two queues take precedence over the lower priority queues if there are packets in them and they have a sufficient credit status.

First, let's take a look at the schedule's first rotation. Q2 is set to be served in the first cycle since DWRR partitions queues with the same priority level. Quantum 2 has a value of 200. Since its credit is at level 200, it can process one packet of size 200 bytes. Since Q2's deficit counter is now at 0, it will not receive service until the next scheduling cycle, as was previously described.

Q3 is the next high-priority queue to be serviced. Due to its quantum nature, this queue schedules one packet but then goes into a negative credit state. The credit status of Q3 is 100%. The deficit counter drops to -900 after 1000 bytes are removed from the queue.

Q2 gets revisited in the next scheduling round because it has a high priority and contains packets. Figure 6.21 depicts a queue with a size of 200 bytes, where the quantum is the same value.

Q3 is currently in a negative credit situation, which is a really fascinating development. Since the deficit counter is less than 1, no scheduling cycles are allocated to it. As a direct consequence, scheduling turn 2 has been moved to the lower priority queues.

During the scheduling cycle, one 1000-byte packet from queue Q1 is processed and the queue's deficit counter is reset to zero since Q1 has a higher priority than queue 0. In turn 2 Q0 is scheduled next, and it clears 800 bytes before entering a negative credit position.

Now we have a situation where two queues, queue 0 for the lowest priority and queue 3 for the highest, are both in a negative credit position. If the current queue states are respected, then queue 1 will receive scheduling service on the following turn, as it is not currently in a negative credit state.

Due to the lack of packets in queue 2, the weight assigned to queue 2 can be used by the other two queues in a negative credit condition to update their credit statuses. In order to improve a queue's credits, certain scheduling rounds will include skipping the queue altogether until the deficit number is larger than zero.

Since the penalized queues can re-use the available weights that aren't being used by other queues, this filling of credits occurs faster if there are no packets in any other queues with a positive credit condition.

How can queues with the same priority level be scheduled in the most effective manner? A suitable architecture would likely

use a mix of strict-high and DWRR scheduling for equivalence classes of queues.

This is an excellent design for the network's core in the event of a loss and subsequent reconvergence, when the strict-priority queue may require more bandwidth for a limited period. Instead, real-time traffic should be throttled at the network's periphery, and a mix of strict-high and DWRR scheduling is probably the best approach. However, if there are many high-priority queues, some form of management, such as DWRR, is required to prevent resource conflicts. The following is an advantage of PB-DWRR:

The shortcomings of existing queuing algorithms have been addressed in PB- DWRR. Most of the innovations made by these algorithms are reflected in it, including byte deficit scheduling and priority tiers.

The following is a restriction of PB-DWRR:

Since it is not a standard, not all routers and switches that use it will behave in the same way. Hardware, resources, and the vendor's actual implementation all have significant effects on how well PB- DWRR functions.

Results from a Study of the Most Effective Queuing Method:

PB-DWRR is now the most widely used queuing and scheduling method.

PB-DWRR is built on the foundation of what came before it, which is why it has been so well received in the industry.

In this article, we covered the basics of PB- DWRR. However, vendors may do things somewhat differently in practice. Any method for queuing and scheduling will inevitably rely on a limited set of materials and apparatus.

1.7 Terminal questions:

1. What is network hardware? Give some common examples of network hardware devices.
2. What is the role of network software in a computer network? Explain the different types of network software?
3. Give some examples of network services commonly used in businesses.
4. What is hardware protection in the context of network security?
5. What are some common methods used for hardware protection?
6. What are two common strategies to protect computer systems from viruses and malware?
7. How can I enhance the security of my devices?
8. What is a VPN and how does it work?
9. What are the common security risks and threats in data centre environments?
10. How can network segmentation contribute to data centre security?
11. What is the role of firewalls in securing enterprise networks?

12. How can intrusion detection and prevention systems (IDPS) enhance data centre security?
13. What are the key considerations for securing a wireless LAN?
14. How can you secure the management plane of a router?
15. What is the purpose of VPN (Virtual Private Network) in router security?
16. What is the purpose of the Round Robin scheduling algorithm?
17. How does Weighted Fair Queuing (WFQ) differ from other scheduling algorithms?
18. What are the performance metrics used to evaluate queuing and scheduling algorithms?
19. Why is it important to consider queuing and scheduling algorithms in web servers or network systems?

Phone 919782065400.
Email: dryogeshsharma@kluniversity.in
https:kluniversity.in

Chapter Two - Key Management and Distribution.

Key distribution is the function that delivers a key to two parties who wish to exchange secure encrypted data. Some sort of mechanism or protocol is needed to provide for the secure distribution of keys.

Key management refers to managing cryptographic keys within a cryptosystem. It deals with generating, exchanging, storing, using, and replacing keys as needed at the user level. A key management system will also include key servers, user procedures and protocols, including cryptographic protocol design.

Symmetric schemes require both parties to share a common secret key. Public key schemes require parties to acquire valid public keys.

Cryptographic, protocol, and management issues are all involved in the complicated topics of cryptographic key management and distribution.

2.1 Key distribution using Symmetric Encryption.

Symmetric encryption requires usage of common key by both the parties sharing of common key was also problem here. And that key must be protected from access by others. Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key. This is one of the most critical areas in security systems - on many occasions systems have been broken, not because of a poor encryption algorithm, but because of poor key selection or management.

Below we look into the methods available for the distribution of symmetric key (session key) by using symmetric encryption:

Assumption is A and B are the parties who want to have the communication.

1. A can select key and physically deliver to B. (Awkward)
2. Third party can select & deliver key to A & B. (Awkward)
3. If A & B have communicated previously can use previous key to encrypt a new key. (Risk of old key usage)
4. If A & B have secure communications with a third-party C, C can relay key between A & B. (recommended usage of variety of it)

The strength of any cryptographic system thus depends on the key distribution technique. For two parties A and B, key distribution can be achieved in several ways:

Physical delivery (1 & 2) is simplest - but only applicable when there is personal contact between recipient and key issuer. This is fine for link encryption where devices & keys occur in pairs but does not scale as number of parties who wish to communicate grows (see next slide). 3 is mostly based on 1 or 2 occurring first, and also suffers that if an attacker ever succeeds in gaining access to one key, then all subsequent keys will be revealed.

A third party, whom all parties trust, can be used as a trusted intermediary to mediate the establishment of secure communications between them (4). Must trust intermediary not to abuse the knowledge of all session keys. As the number of parties grow, some variant of 4 is only practical solution to the huge growth in number of keys potentially needed.

The below hierarchy of keys shows the relationship between various keys. The main categories are mentioned below.

Master Key

It is used to encrypt session keys.

It is shared by user & key distribution centre.

Session key

It is a temporary key.

It is used for encryption of data between users.

It is for one logical session then discarded.

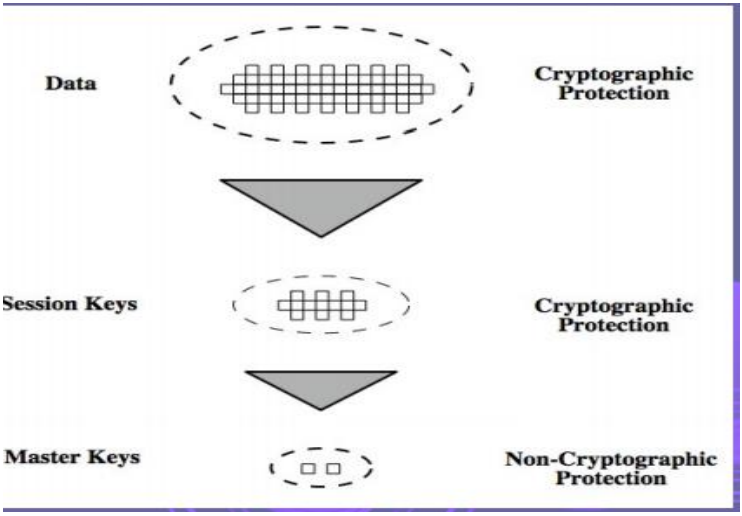

Figure2.1: Key hierarchy. (Copy right is reserved to William Stallings.)

Key distribution scenario:

1. A requests from the KDC a session key to protect a logical connection to B. The message includes the identity of A and B and a unique nonce N1.
2. The KDC responds with a message encrypted using Ka that includes a one-time session key Ks to be used for the session, the original request message to enable A to match response with appropriate request, and info for B.
3. A stores the session key for use in the upcoming session and forwards to B the information from the KDC for B, namely, E(Kb, [Ks || IDA]). Because this information is encrypted with Kb, it is protected from eavesdropping.

At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. Two additional steps are desirable:

4. Using the new session key for encryption B sends a nonce N2 to A.

5. Also using Ks, A responds with f(N2), where f is a function that performs some transformation on N2 (eg. adding one). These steps assure B that the original message it received (step 3) was not a replay. Note that the actual key distribution involves only steps 1 through 3 but that steps 4 and 5, as well as 3, perform an authentication function.

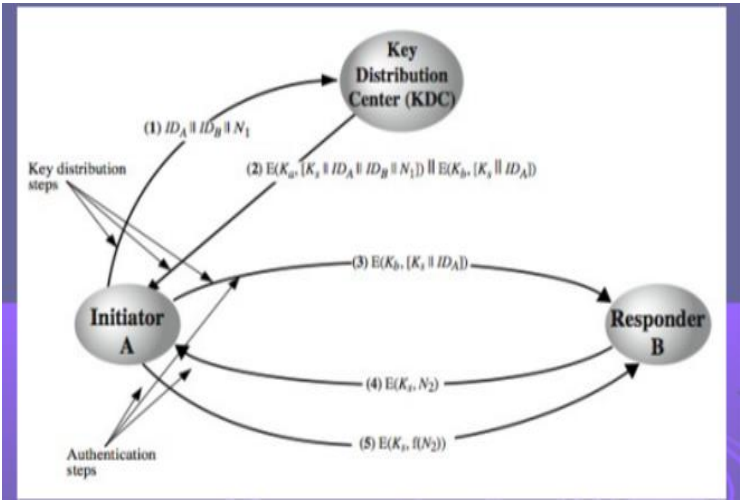

Figure2.2: Key distribution using KDC. (Copy right is reserved to William Stallings.)

Key distribution issues:

Some of the major issues associated with the use of Key Distribution Centers (KDC's) are to be noted.

For very large networks, a hierarchy of KDCs can be established. For communication among entities within the same local domain, the local KDC is responsible for key distribution. If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a (hierarchy of) global KDC(s)

For each new connection-oriented session, a new session key should be utilized to strike a compromise between security and

effort. A new session key for a connectionless protocol is only utilized for a predetermined amount of time or for a predetermined number of transactions.

An automated key distribution approach provides the flexibility and dynamic characteristics needed to allow several terminal users to access several hosts and for the hosts to exchange data with each other, provided they trust the system to act on their behalf.

The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized.

2.2 Key distribution using Asymmetric Encryption.

Public key and private keys are to be used in Asymmetric encryption. Public key cryptosystems are inefficient so almost never use for direct data encryption, rather use to encrypt secret keys for distribution.

Simple Secret Key Distribution:

Merkle proposed this very simple scheme that allows secure communications and no keys before/after exist.

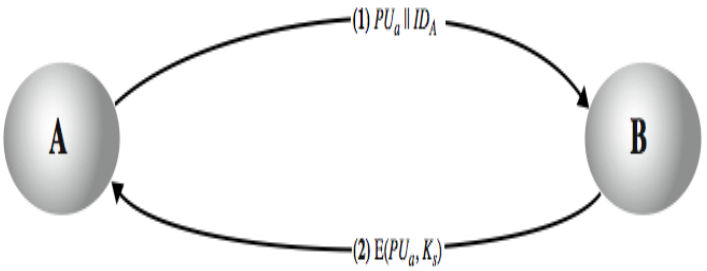

Figure2.3: Simple Key distribution using Public Keys. (Copy right is reserved to William Stallings.)

Man-in-the-Middle Attack:

This very simple scheme is vulnerable to an active man-in-the-middle attack.

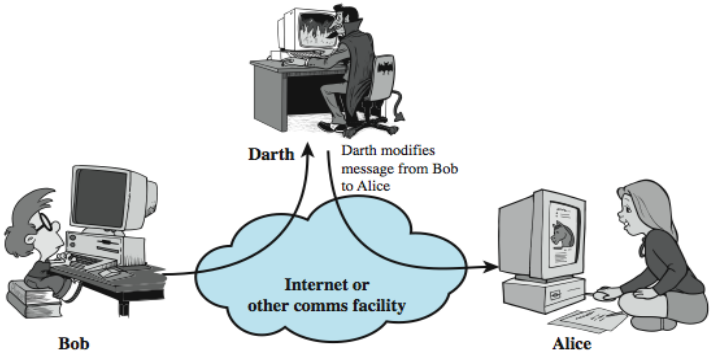

Figure2.4: Man in the middle attack. (Copy right is reserved to William Stallings.)

The following steps occur to ensure both confidentiality and authentication:

- A uses B's public key to encrypt a message to B containing an identifier of A (IA) and a nonce (N1), which is used to identify this transaction uniquely.
- B sends a message to A encrypted with PUa and containing A's nonce (N1) as well as a new nonce generated by B (N2). Because only B could have decrypted message (1), the presence of N1 in message (2) assures A that the correspondent is B.
- A returns N2, encrypted using B's public key, to assure B that its correspondent is A.
- A selects a secret key Ks and sends M = E(PUb, E(PRa, Ks)) to B. Encryption with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.

- B computes D(PUa, D(PRb, M)) to recover the secret key.

The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

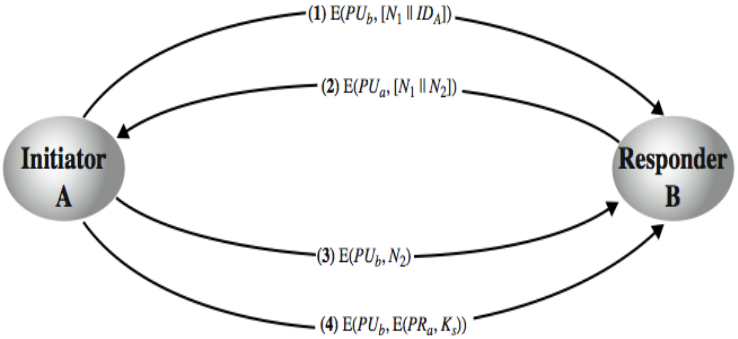

Figure2.5: Key distribution using Public Encryption (Copy right is reserved to William Stallings.).

2.3 Distribution of Public Keys.

Several techniques are proposed for the distribution of public keys, those can mostly be grouped into the categories mentioned below.

1. Public announcement
2. Publicly available directory
3. Public-key authority
4. Public-key certificates

2.3.1 Public announcement:

Users share their public keys with recipients by broadcasting or make them available to the broader community. For example, they may attach PGP keys to email messages or post them on news groups or email lists. However, a significant vulnerability in this approach is the possibility of forgery. Anyone can create a key pretending to be someone else and disseminate it, allowing

them to impersonate the claimed user until the forgery is detected. The concept of public-key encryption relies on the fact that the public key is meant to be public, enabling any participant to share their public key with others or broadcast it to the community. Nevertheless, the risk of forgery remains a major concern, as malicious individuals can take advantage of this openness to deceive others.

2.3.2 Publicly Available Directory:

Enhanced security can be achieved by maintaining a dynamic directory of public keys accessible to the public. For this purpose, a reliable body or organization must be responsible for maintaining and distributing the public directory. Although this approach offers more security than individual public statements, it is not entirely immune to the risks of fraud or sabotage.

To achieve higher security, keys can be registered with a public directory. To be deemed trustworthy, the directory must possess certain properties:

- It contains {name, public-key} entries for each participant.
- Participants securely register their keys with the directory.
- Participants have the flexibility to replace their keys at any time.
- The directory is periodically published.
- The directory is accessible electronically.

However, despite these measures, the directory remains vulnerable to tampering or forgery.

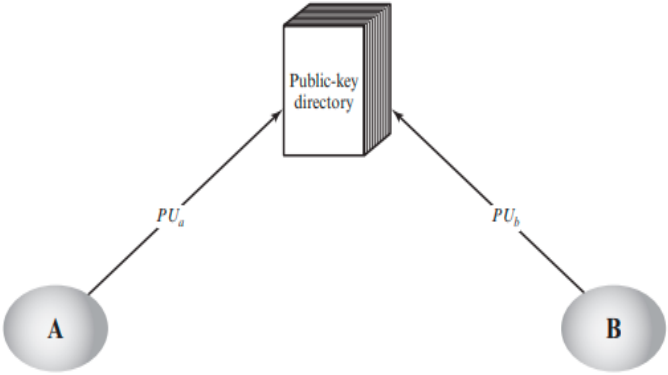

Figure 2.6: usage of public key directories by parties A and B. (Copy right is reserved to William Stallings.)

2.3.3 Public key Authority

"Public-Key Authority" serves as an example of a common protocol exchange. The scenario continues to presume that a central authority has a dynamic directory of all participants' public keys. Additionally, only the authority is aware of the associated private key, whereas each participant reliably knows the authority's public key. The steps of the protocol are detailed in the text. There are a total of seven messages needed. However, because both A and B can cache each other's public keys for later use, the first four messages only need to be sent occasionally. To maintain currency, a user should periodically request new copies of the public keys of its correspondents.

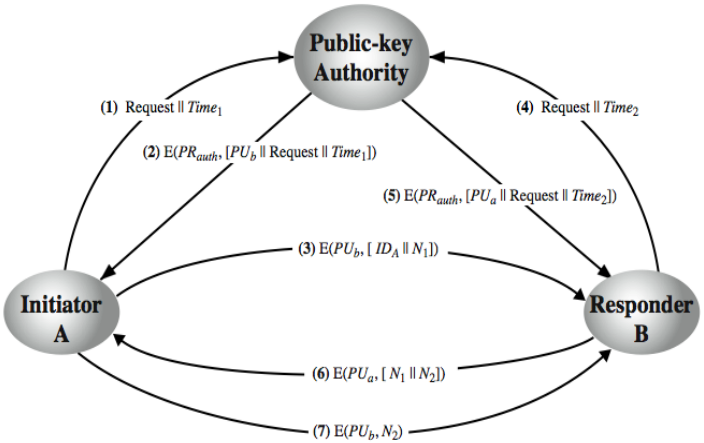

Figure 2.7: usage of public key authority. (Copy right is reserved to William Stallings.)

- Stronger security can be achieved by tightening control over distribution of keys from directory.
- has properties of directory and requires users to know public key for the directory.
- then users interact with directory to obtain any desired public key securely
- does require real-time access to directory when keys are needed.
- may be vulnerable to tampering.

Public key Authority bottleneck

The public-key authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public

key for every other user that it wishes to contact. As before, the directory of names and public keys maintained by the authority is vulnerable to tampering.

2.3.4 Public-Key Certificates

Each participant submits an application to the certificate authority, including a public key and a certificate request. The application process must be conducted in person or through a secure communication method. The authority issues participant A a certificate (CA). A can then share this certificate with other participants, who can read it and verify its authenticity by confirming the certificate authority's signature. The signature confirms that the certificate indeed originates from the certificate authority since it can only be decrypted with the authority's public key.

The following scenario involves a timestamp. Suppose an opponent manages to obtain A's secret key. In response, A generates a new private/public key pair and requests a fresh certificate from the certificate authority. However, the adversary is attempting to deceive B by presenting the old certificate. In this situation, the attacker can access messages that B encrypts using the compromised old public key.

To understand this situation better, consider it analogous to losing a credit card. When a credit card is lost, the owner cancels the card number. Nevertheless, until all potential recipients are informed that the previous card is no longer valid, they remain at risk. Similarly, the timestamp acts as an expiration time. A certificate is considered expired if it reaches a certain age.

- certificates allow key exchange without real-time access to public-key authority.

- a certificate binds **identity** to **public key** usually with other info such as period of validity, rights of use etc
- with all contents **signed** by a trusted Public-Key or Certificate Authority (CA) can be verified by anyone who knows the public-key authorities public-key.

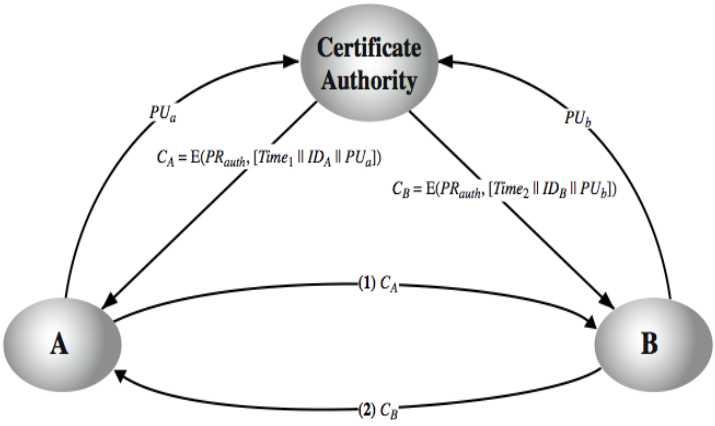

Figure 2.8: usage of certificate authority. (Copy right is reserved to William Stallings.)

2.4 Public Key Infrastructure.

RFC 4949 (Internet Security Glossary) defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.

The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group has been the driving

force behind setting up a formal (and generic) model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet.

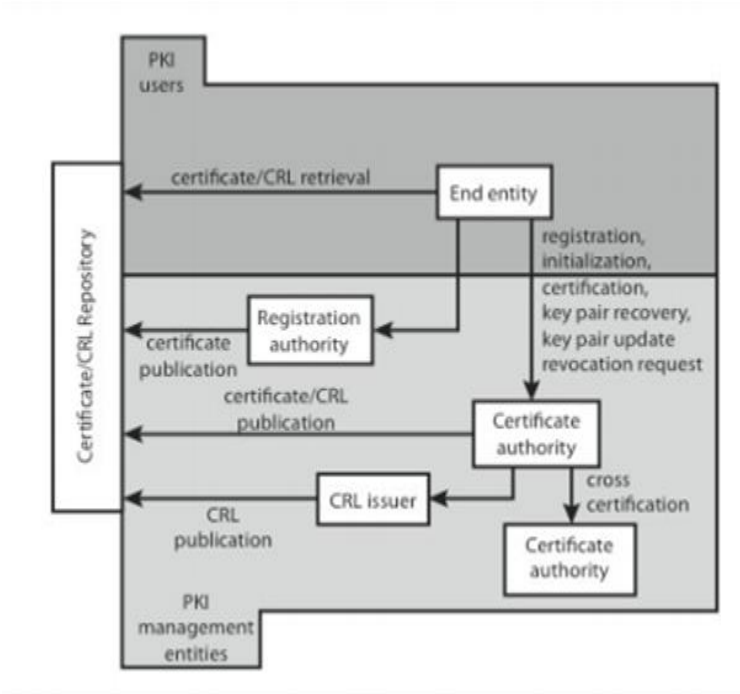

Figure 2.9: Public Key Infrastructure. (Copy right is reserved to William Stallings.)

The main elements are:

1. End entity
2. Certification authority (CA)
3. Registration authority (RA)
4. CRL issuer
5. Repository

PKIX identifies several management functions that potentially need to be supported by management protocols, as shown in above Figure.:

• **Registration:** whereby a user first makes itself known to a CA, prior to issue of a certificate(s) for that user. It usually involves some off-line or online procedure for mutual authentication.

• **Initialization:** to install key materials that have the appropriate relationship with keys stored elsewhere in the infrastructure.

• **Certification:** process where a CA issues a certificate for a user's public key and returns it to the user's client system and/or posts it in a repository.

• **Key pair recovery:** a mechanism to recover the necessary decryption keys when normal access to the keying material is no longer possible.

• **Key pair update:** key pairs need to be updated and new certificates issued.

• **Revocation request:** when authorized person advises need for certificate revocation, e.g. private key compromise, affiliation change, name change.

• **Cross certification:** when two CAs exchange information used in establishing a cross-certificate, issued by one CA to another CA that contains a CA signature key used for issuing certificates.

2.5 X.509 Certificates.

Introduction:

- Recommendation of ITU-D and series of X.500.

- X.509 defines a framework for the provision of authentication services by the X.500 directory to its users.
- Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority.
- X.509 defines alternative authentication protocols based on the use of public-key certificates.
- X.509 certificate format is used in S/MIME.
- X.509 is based on the use of public-key cryptography and digital signatures.

2.5.1 X.509 Public-Key Certificate Use:

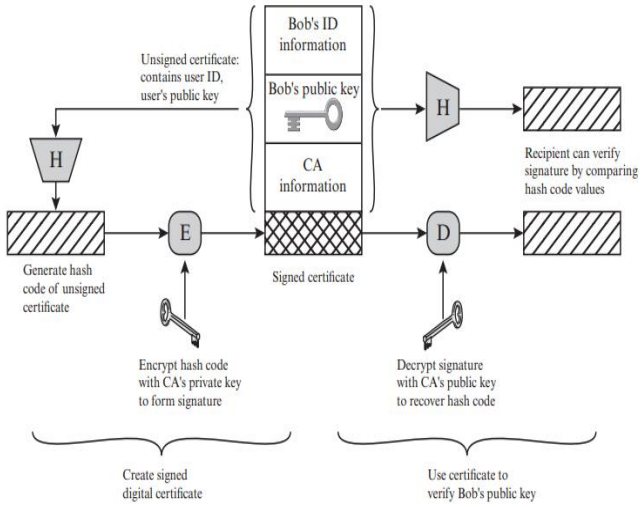

Figure 2.10: Usage of Public key certificate. (Copy right is reserved to William Stallings.)

The certificate for Bob's public key includes unique identifying information for Bob, Bob's public key, and identifying information about the CA, plus other information as explained subsequently. This information is then signed by computing a hash value of the information and generating a digital signature using the hash value and the CA's private key.

The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.

2.5.2 X.509 Certificate Format:

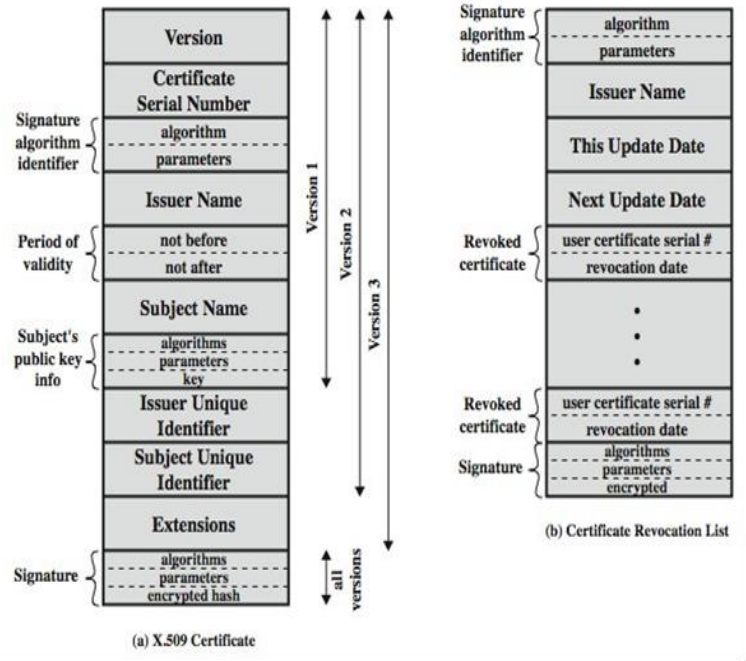

Figure 2.10: X.509 Certificate format with different versions (Copy right is reserved to William Stallings.)

An X.509 certificate contains details about both the recipient's identity and the entity that issued it. The standard information within an X.509 certificate includes the following:

Version: Indicates the X.509 version applicable to the certificate and specifies the data that the certificate must contain.

Serial number: Assigned by the Certificate Authority (CA) to distinguish the certificate from others.

Algorithm information: Specifies the signature algorithm used by the issuer to sign the certificate.

Issuer distinguished name: Refers to the name of the entity that issues the certificate, typically the CA.

Validity period: Includes the start and end dates during which the certificate can be trusted and considered valid.

Subject distinguished name: Refers to the name of the entity for which the certificate is intended.

Subject public key information: Contains the public key associated with the entity's identity.

Extensions (optional): These have unique IDs expressed as an object identifier (OID) set of values. Extensions may be rejected if they are not recognized or contain un processable information.

Obtaining a certificate:

User certificates generated by a CA have the characteristics that any user with access to the public key of the CA can verify the user public key that was certified, and no party other than the certification authority can modify the certificate without this being detected. Because certificates are unforgeable, they can be placed in a directory without the need for the directory to make special efforts to protect them.

2.5.3 CA Hierarchy

If both parties utilize the same Certification Authority (CA), they possess the CA's public key and can verify the certificates

of others. However, in scenarios where there is a large user community, it might not be feasible for all users to rely on a single CA. In such cases, multiple CAs could be established, and each CA securely provides its public key to a specific subset of users.

To ensure a secure connection between the CAs used by the two parties, a chain of certifications is formed using client and parent certificates. All these CA certificates need to be available in the directory, and users must understand their interconnections to follow a path to another user's public-key certificate. X.509 proposes arranging CAs in a hierarchical structure, simplifying navigation. It is assumed that each client trusts the certificates issued by its parent CA.

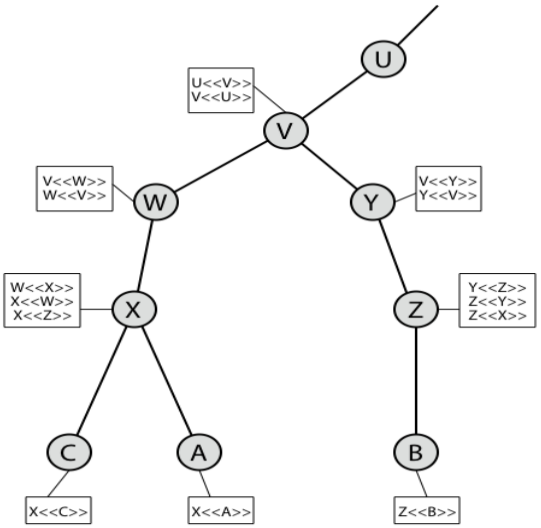

Figure 2.11: CA Hierarchy (Copy right is reserved to William Stallings.)

A certificate comes with a designated validity period, and usually, a new certificate is issued just before the expiration of the old one. However, there might be occasions when it becomes

necessary to revoke a certificate before its natural expiration due to various reasons as mentioned earlier.

To facilitate such revocations, every Certification Authority (CA) is required to maintain a list known as the Certificate Revocation List (CRL). This list contains details of all the certificates that have been revoked by the CA but have not yet expired. Each CRL posted to the directory is signed by the issuer and contains essential information, including the issuer's name, the creation date of the list, the scheduled issuance date of the next CRL, and entries for each revoked certificate. Each entry consists of the serial number of the revoked certificate and the date of revocation. As serial numbers are unique within a CA, they are sufficient for identifying the specific certificates.

When a user receives a certificate in a message, it becomes crucial for the user to check whether the certificate has been revoked. This check involves verifying against the directory's CRL each time a certificate is received. However, in practice, this verification process is often neglected or not consistently performed.

Key and Policy Information:

A certificate policy refers to a specified collection of rules that determine the suitability of a certificate for a specific community and/or type of application, encompassing shared security requirements.

It includes:

1. Authority key identifier.
2. Subject key identifier.
3. Key usage.
4. Private-key usage period.
5. Certificate policies.
6. Policy mappings.

2.6 User Authentication Using Symmetric Encryption.

Symmetric encryption is a method of encrypting and decrypting data using the same key. It offers relatively easy implementation

but necessitates keeping the key secret. User authentication involves verifying a user's identity, typically by requiring something they know, have, or are, like a password, security token, or fingerprint.

To apply symmetric encryption for user authentication, the client and server must share a secret key. The client encrypts a challenge message with the key and sends it to the server, which decrypts it to verify the client's identity. An example of this process includes the client requesting a service from the server, and the server responding with an encrypted challenge message, which the client decrypts and returns for authentication.

While symmetric encryption can securely authenticate users, safeguarding the shared secret key is crucial to prevent unauthorized access. Security considerations involve using a strong, unique key generated by a secure random number generator, avoiding plaintext storage, and periodic key rotation to reduce risks.

The strategy employs a trusted Key Distribution Center (KDC), where parties share a master key. The KDC generates session keys for secure connections and distributes them using the master keys.

For one-way authentication, the recipient's electronic mailbox stores forwarded email until accessed. The recipient seeks assurance of the sender's authenticity.

The Needham-Schroeder Protocol, an original third-party key distribution protocol mediated by KDC, facilitates secure session key distribution between parties A and B.

Protocol overview is:

1. A->KDC: IDA || IDB || N1
2. KDC -> A: E(Ka,[Ks||IDB||N1|| E(Kb,[Ks||IDA])])

3. A -> B: E(Kb, [Ks||IDA])
4. B -> A: E(Ks, [N2])
5. A -> B: E(Ks, [f(N2)])

However, it is vulnerable to replay attacks if an old session key is compromised.

As a consequence, message 3 could be resent, persuading B that it is communicating with A.

Denning proposed modifications to overcome this drawback by incorporating the following measures:

• Utilizing timestamps in steps 2 & 3 (Denning 81).
• Introducing an additional nonce (Neuman 93).

Denning Approach

1. $A \rightarrow KDC: \quad ID_A \| ID_B$
2. $KDC \rightarrow A: \quad E(K_a, [K_s \| ID_B \| T \| E(K_b, [K_s \| ID_A \| T])])$
3. $A \rightarrow B: \quad E(K_b, [K_s \| ID_A \| T])$
4. $B \rightarrow A: \quad E(K_s, N_1)$
5. $A \rightarrow B: \quad E(K_s, f(N_1))$

T represents a timestamp that provides confidence to both A and B that the session key has been recently generated.

• Therefore, both A and B are assured that the key distribution is a new and fresh exchange.
• To verify the timeliness, A and B can check for the following:

This refinement of the Key Distribution Center (KDC) is employed to enhance email security.

However, since B is not online, steps 4 and 5 can be omitted.

Protocol becomes:

1. A->KDC: IDA || IDB || N1.
2. KDC -> A: E(Ka, [Ks||IDB||N1 || E(Kb,[Ks||IDA])]).
3. A -> B: E(Kb, [Ks||IDA]) || E(Ks, M).

This protocol offers encryption and some authentication capabilities but does not protect against replay attacks.

2.7 User Authentication using Asymmetric Encryption.

User authentication is the process of verifying a user's identity. Asymmetric encryption offers a means for user authentication by storing the user's public key on a server. To authenticate themselves, the user sends a message encrypted with their private key to the server. The server can then decrypt the message using the user's public key, thus confirming the user's identity.

Asymmetric encryption involves two keys: a public key and a private key. The private key is kept secret, while the public key can be openly shared.

There are two primary methods of user authentication using asymmetric encryption:

• Mutual authentication: Both the user and the server authenticate each other.

• One-way authentication: Only the user authenticates themselves to the server.

Mutual authentication using denning protocol:

1. $A \rightarrow AS: \quad ID_A \| ID_B$
2. $AS \rightarrow A: \quad E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T])$
3. $A \rightarrow B: \quad E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T]) \| E(PU_b, E(PR_a, [K_s \| T]))$

The central system in this context is known as an authentication server (AS). However, its role does not involve secret-key

distribution. Instead, the AS provides public-key certificates. The session key is selected and encrypted by party A, ensuring that the AS does not pose a risk of exposing it. The use of timestamps further safeguards against replay attacks involving compromised keys.

Some other protocol proposed by woo and lam:

1. $A \rightarrow KDC: \quad ID_A \| ID_B$
2. $KDC \rightarrow A: \quad E(PR_{auth}, [ID_B \| PU_b])$
3. $A \rightarrow B: \quad E(PU_b, [N_a \| ID_A])$
4. $B \rightarrow KDC: \quad ID_A \| ID_B \| E(PU_{auth}, N_a)$
5. $KDC \rightarrow B: \quad E(PR_{auth}, [ID_A \| PU_a]) \| E(PU_b, E(PR_{auth}, [N_a \| K_s \| ID_B]))$
6. $B \rightarrow A: \quad E(PU_a, [E(PR_{auth}, [(N_a \| K_s \| ID_B)]) \| N_b])$
7. $A \rightarrow B: \quad E(K_s, N_b)$

The revised protocol of the above one is mentioned below

1. $A \rightarrow KDC: \quad ID_A \| ID_B$
2. $KDC \rightarrow A: \quad E(PR_{auth}, [ID_B \| PU_b])$
3. $A \rightarrow B: \quad E(PU_b, [N_a \| ID_A])$
4. $B \rightarrow KDC: \quad ID_A \| ID_B \| E(PU_{auth}, N_a)$
5. $KDC \rightarrow B: \quad E(PR_{auth}, [ID_A \| PU_a]) \| E(PU_b, E(PR_{auth}, [N_a \| K_s \| ID_A \| ID_B]))$
6. $B \rightarrow A: \quad E(PU_a, [N_b \| E(PR_{auth}, [N_a \| K_s \| ID_A \| ID_B])])$
7. $A \rightarrow B: \quad E(K_s, N_b)$

Disadvantages of existing protocols:

Various approaches necessitate either the sender knowing the recipient's public key (for confidentiality), the recipient knowing the sender's public key (for authentication), or both (for confidentiality and authentication). Additionally, the public-key algorithm may need to be applied once or twice to a potentially lengthy message.

2.8 Kerberos.

Kerberos is a robust network authentication protocol that ensures secure user authentication and encrypted communication across networks. Developed by MIT as part of Project Athena, Kerberos addresses the challenges of securely authenticating users in distributed computing environments.

Key Components: Kerberos involves three primary components: the client, the server, and the Key Distribution Center (KDC).

Authentication Process: The Kerberos authentication process follows these steps:

a. Authentication Request: The client sends an authentication request to the KDC, providing its identity.

b. Ticket Granting Ticket (TGT): Upon receiving the request, the KDC verifies the client's identity and issues a Ticket Granting Ticket (TGT) containing the client's identity and a session key.

c. Ticket: The client presents the TGT to the KDC to request a service ticket for a specific server.

d. Session Key: The KDC issues a service ticket, encrypted with the server's key, along with a session key. The client receives the service ticket and decrypts it using its session key.

e. Service Authentication: The client presents the decrypted service ticket to the server. The server decrypts the service ticket using its key and verifies the client's identity.

f. Session Establishment: If the client's identity is verified, the server and client establish a secure session using the session key.

Ticket Lifetime and Renewal: Kerberos tickets have limited lifetimes. After expiration, the client must request a new ticket from the KDC. However, Kerberos supports ticket renewal to extend the validity without reauthentication.

Encryption and Security: Kerberos utilizes symmetric key cryptography to secure the authentication process and subsequent communication between the client and server. Shared secret keys are securely stored within the KDC and the client's devices.

Kerberos Realms: Kerberos uses realms to define administrative domains, each with its own KDC. Cross-realm authentication is possible through established trust relationships between realms.

Applications: Kerberos finds extensive use in systems and applications such as Microsoft Active Directory for Windows domain authentication, secure shell (SSH) protocol, and network file systems (NFS).

Security Considerations: While Kerberos offers robust authentication and encryption mechanisms, safeguarding the KDC and shared secret keys is crucial. Regular key updates, secure key storage, and protection against replay attacks are essential considerations.

Kerberos Version 4 Flow:

Authentication Phase:

a. The client requests a TGT from the AS.

b. The request includes the client's username and the TGS's name.

c. The AS verifies the client's identity, checks the user's password, and issues a TGT encrypted with the client's password.

Ticket Granting Phase:

a. The client requests a service ticket from the TGS.

b. The request includes the client's TGT and the desired network service's name.

c. The TGS decrypts the TGT using the client's password and verifies its authenticity.

d. If valid, the TGS issues a service ticket encrypted with the client's session key, along with a copy of the session key encrypted with the service's secret key.

Service Request Phase:

a. The client requests access to the network service, presenting the service ticket obtained from the TGS.

b. The service decrypts the service ticket using its secret key to verify its authenticity.

c. If the service ticket is valid, the client gains access to the requested network service.

Kerberos version 4 employs symmetric key cryptography and timestamps for secure communication and prevention of replay attacks. However, it lacks forward secrecy and support for secure cross-realm authentication, which were improved in later versions of Kerberos.

(a) Authentication Service Exchange: to obtain ticket-granting ticket

(1) C → AS: $ID_c \| ID_{tgs} \| TS_1$

(2) AS → C: $E_{K_c}\left[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}\right]$

$$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2\right]$$

(b) Ticket-Granting Service Exchange: to obtain service-granting ticket

(3) C → TGS: $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C: $E_{K_{c,tgs}}\left[K_{c,v} \| ID_v \| TS_4 \| Ticket_v\right]$

$$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2\right]$$

$$Ticket_v = E_{K_v}\left[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4\right]$$

$$Authenticator_c = E_{K_{tgs}}\left[ID_C \| AD_C \| TS_3\right]$$

(c) Client/Server Authentication Exchange: to obtain service

(5) C → V: $Ticket_v \| Authenticator_c$

(6) V → C: $E_{K_{c,v}}\left[TS_5 + 1\right]$ (for mutual authentication)

$$Ticket_v = E_{K_v}\left[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4\right]$$

$$Authenticator_c = E_{K_{c,v}}\left[ID_C \| AD_C \| TS_5\right]$$

Kerberos 4 Overview

Figure 2.12: Dialog exchange of Kerberos Version4 (Copy right is reserved to William Stallings.)

The concept of *realm* can be explained as follows.

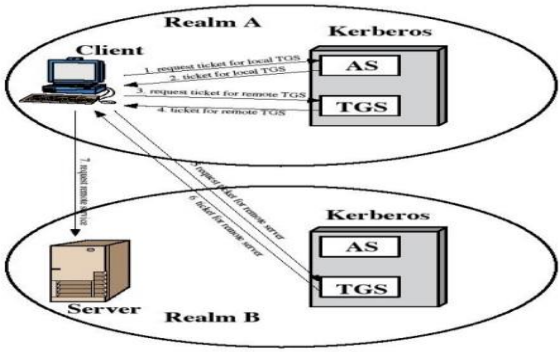

Figure 2.13: Inner realm. (Copy right is reserved to William Stallings.)

Kerberos V5 offers a range of security features, ensuring robust protection for network communications:

Strong Authentication: The client and server applications mutually authenticate using a shared secret key.

Confidentiality: Data exchanged between the client and server is encrypted, preventing unauthorized access.

Integrity: Data exchanged between the client and server is verified to guarantee it remains unaltered.

Kerberos V5 is a widely adopted and dependable authentication protocol, making it an excellent choice for applications requiring data protection, confidentiality, and integrity during communications.

Advantages of using Kerberos V5 include:

Enhanced Security: Kerberos V5 provides robust authentication, confidentiality, and integrity, bolstering overall network security.

Reduced Complexity: Kerberos V5 simplifies user account and password management, streamlining administrative tasks.

Scalability: Kerberos V5 is designed to support large networks, ensuring seamless operations as networks grow in size.

Improved Performance: By minimizing the number of network round trips required for user authentication, Kerberos V5 enhances network performance.

If you seek a secure and reliable authentication protocol for your network, Kerberos V5 is a highly recommended option. Its capabilities ensure a strong defense against unauthorized access and tampering of data, making it an ideal choice for safeguarding sensitive communications.

2.9 Terminal Questions.

1. Describe KDC.
2. List out Keys in keys Hierarchy.
3. Analyze How many keys are required if an organization with n users want to communicate securely using symmetric keys.
4. Describe Public announcement.
5. Mention about the content of Public available Directory.
6. Mention the purpose of Certificate authority.
7. Define process of caching.
8. Describe Certificate.
9. What is X.509.
10. Define Certification revocation.
11. How are X.509 certificates used?
12. How key compromise risk associated with X.509 certificates show the effect?
13. What is certificate revocation?
14. What is certificate pinning?
15. What is Authentication?
16. Difference between one way authentication and mutual authentication
17. What is replay attack?
18. What is Authentication?
19. 2. Difference between one way authentication and mutual authentication.
20. How is asymmetric encryption used for user authentication?
21. What are the benefits of using asymmetric encryption for user authentication?
22. What are the challenges of using asymmetric encryption for user authentication?
23. What is Authentication service?

24. Explain client server environment.
25. Difference between Kerberos Version4 and Kerberos Version5
26. What is Realm?
27. What is Kerberos?
28. How does Kerberos work?
29. What are the components of the Kerberos authentication protocol?

Phone: 9966302375.
Email: ramaiah.challa@gmail.com
https:kluniversity.in

Chapter Three – Web Security.

3.1 Web security and issues.

3.1.1 Web Security Threats

These dangers can be categorized in terms of passive and active attacks. Eavesdropping on network traffic between a browser and a server and getting access to data on a website that is supposed to be limited are examples of passive attacks. Active attacks include spoofing other users, tampering with communications as they are being sent between a client and a server, and changing data on a website.

The threat's location, such as the Web server, browser, or network traffic between the server and browser, can also be used to categorize web security concerns. Computer system security concerns include those relating to server and browser security.

	Threats	Consequences	Countermeasures
Integrity	• Modification of user data • Trojan horse browser • Modification of memory • Modification of message traffic in transit	• Loss of information • Compromise of machine • Vulnerability to all other threats	Cryptographic checksums
Confidentiality	• Eavesdropping on the net • Theft of info from server • Theft of data from client • Info about network configuration • Info about which client talks to server	• Loss of information • Loss of privacy	Encryption, Web proxies
Denial of Service	• Killing of user threads • Flooding machine with bogus requests • Filling up disk or memory • Isolating machine by DNS attacks	• Disruptive • Annoying • Prevent user from getting work done	Difficult to prevent
Authentication	• Impersonation of legitimate users • Data forgery	• Misrepresentation of user • Belief that false information is valid	Cryptographic techniques

Tab.3.1 Web Threat Examples

3.1.2 Web Traffic Security Approaches

There are several methods for offering Web security. The numerous strategies that have been taken into consideration are comparable in terms of the services they offer and, to a certain extent, in terms of the mechanisms they employ, but they differ in terms of their applicability and relative positioning within the TCP/IP protocol stack.

This session clearly defines the difference between the security in Network, Transport and Application level.

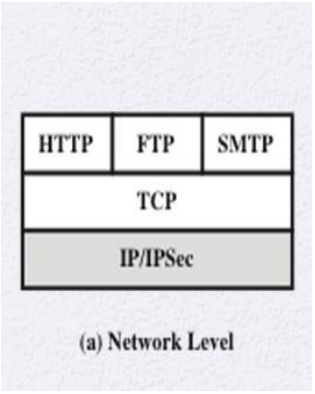

Figure 3.1 Network Level

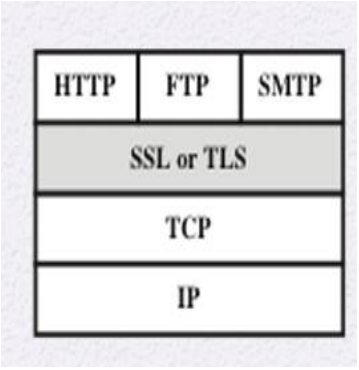

Figure3.2 Transport Level

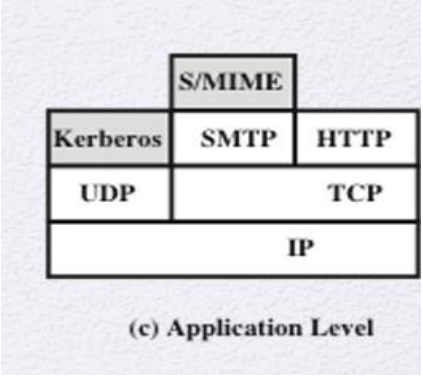

Figure3.3 Application Level

(For all above figures Copy right is reserved to William Stallings.)

a. Network Level Security

IP security (IPsec) is to be used for web security. IPsec has the benefit of being transparent to end users and applications and offering a multipurpose solution. Additionally, IPsec has a filtering feature so that only specific traffic needs to be subject to the overhead of IPsec processing.

b. Transport Level Security

The Secure Sockets Layer (SSL) and the related Internet standard known as Transport Layer Security (TLS) serve as the best examples of this strategy. There are two implementation options at this level. For complete generality, SSL (or TLS) might be made available as a component of the underlying protocol suite and be invisible to applications as a result. TLS can also be included in particular packages. For instance, TLS is a feature that almost all browsers have, and the majority of Web servers have adopted the protocol.

c. Application-Level Security

The particular application includes embedded security services that are application specific.

3.2 Secure Socket Layer Concept

3.2.1 SSL Architecture

TCP is intended to be used by SSL in order to deliver a dependable end-to-end secure service. SSL consists of two layers of protocols rather than one.

Several higher-layer protocols receive fundamental security functions via the SSL Record Protocol. SSL may be used on top of the Hypertext Transfer Protocol (HTTP), which offers the

transfer service for Web client/server interaction. The Handshake Protocol, Change Cypher Spec Protocol, and Alert Protocol are three higher-layer protocols that are included in SSL.

The SSL session and SSL connection are two crucial SSL concepts that are specified in the standard as follows.

• Connection: According to the OSI layering model, a connection is a transport that offers the appropriate kind of service. These partnerships are peer-to-peer ones for SSL. The relationships are fleeting. One session is connected to each connection.

• SSL sessions are connections between a client and a server. With the Handshake Protocol, sessions are produced. A set of cryptographic security settings called sessions are defined and can be shared by many connections. The costly negotiation of new security settings for each connection is avoided by using sessions.

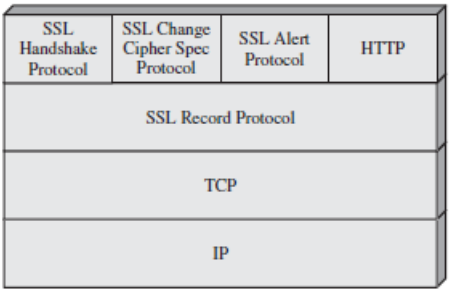

Figure 3.4 SSL Protocol Stack (Copy right is reserved to William Stallings.)

A session state is defined by the following parameters:

- Session identifier
- Peer certificate
- Compression method:
- Cipher spec
- Master secret
- Is resumable

A connection state is defined by the following parameters:

- Server and client random
- Server write MAC secret
- Client write MAC secret
- Server write key
- Client write key
- Initialization vectors
- Sequence numbers

3.2.2 SSL Record Protocol

For SSL connections, the SSL Record Protocol offers the following two services:

- Confidentiality: The Handshake Protocol specifies a shared secret key that is utilised for standard SSL payload encryption.
- Message Integrity: The Handshake Protocol also specifies how to create a message authentication code (MAC) using a shared secret key.

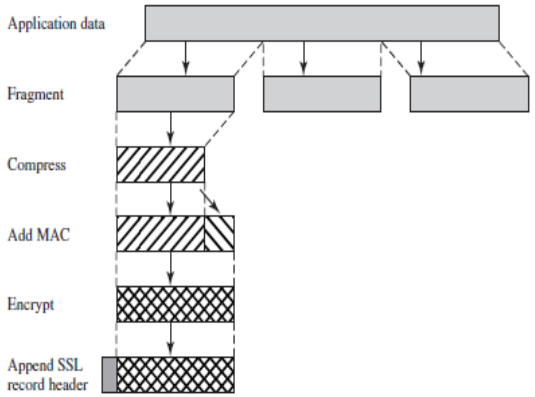

Figure 3.5 SSL Record Protocol Operation (Copy right is reserved to William Stallings.)

The first step is fragmentation. Each upper-layer message is fragmented into blocks of 214 bytes or less (16384 bytes). Compression is then optionally used. Compression must be lossless and can only lengthen material by a maximum of 1024 bytes.1The default compression method in SSLv3 (and the current version of TLS) is null because no compression technique is provided.

Computing a message authentication code over the compressed data is the following stage of processing. An agreed-upon secret key is utilized for this:

```
hash(MAC_write_secret || pad_2||
    hash(MAC_write_secret || pad_1||seq_num ||
    SSLCompressed.type || SSLCompressed.length ||
    SSLCompressed.fragment))
```

The preparation of a header with the following elements is the last stage in the processing of an SSL Record Protocol.

• Content Type (8 bits): The protocol utilised at a higher tier to handle the contained fragment.

• Major Version (8 bits): This number represents the current SSL major version. The value for SSLv3 is 3.

Minor Version (8 bits): Indicates the presence of a minor version. The value for SSLv3 is 0.

• Compressed Length (16 bits): The number of bytes that make up the compressed or plaintext fragment, depending on whether compression is being utilized.

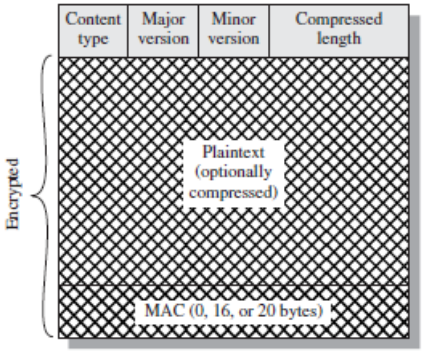

Figure3.6 SSL Record Protocol (Copy right is reserved to William Stallings.)

3.2.3 Change Cipher Spec Protocol

The Change Cipher Spec Protocol, which is part of the SSL suite, is one of three protocols that utilize the SSL Record Protocol. It is considered the most straightforward protocol. This protocol is comprised of a solitary message, consisting of a single byte set to the value 1. The primary objective of this message is to copy the pending state into the current state, thereby updating the cipher suite for the ongoing connection.

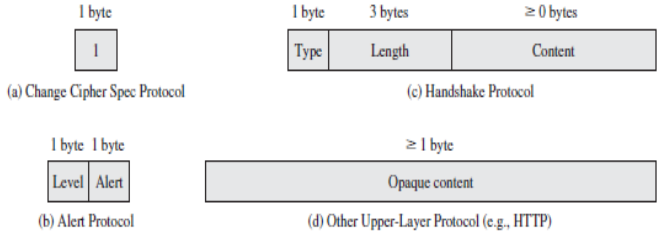

Fig:3.7 SSL Record Protocol Payload

3.2.4 Alert Protocol

The Alert Protocol serves the purpose of transmitting SSL-related alerts to the peer entity. Similar to other applications that employ SSL, alert messages undergo compression and encryption based on the current state's specifications.

In this protocol, every message is composed of two bytes. (The initial byte represents the severity of the message, with a value of either warning (1) or fatal (2). If the severity level is fatal, the SSL protocol promptly terminates the connection. While other connections within the same session can persist,

establishing new connections on the current session is not possible. The second byte carries a code that signifies the specific alert being conveyed. Below, we provide a list of alerts defined by the SSL specification that are always considered fatal.

The following are the alert messages:

- **unexpected_message**
- **bad_record_mac:**
- **decompression_failure:**
- **handshake_failure:**
- **illegal_parameter:**

Additional Alert Messages

- **close_notify**
- **no_certificate**
- **bad_certificate**
- **unsupported_certificate**
- **certificate_revoked**
- **certificate_expired**
- **certificate_unknown**

3.2.5 Handshake Protocol

The Handshake Protocol within SSL represents the most intricate component. Its purpose is to facilitate mutual authentication between the server and client, as well as the negotiation of encryption and MAC algorithms, along with cryptographic keys for safeguarding data transmitted via SSL records. The Handshake Protocol is executed prior to any transmission of application data. Every message comprises three fields:

- One of 10 messages is indicated by the type (1 byte). The specified message types are shown in Table 16.2.
- Length (3 bytes): The message's byte count.
- Content (bytes): The message's identifying information

The initial transaction required to create a logical connection between the client and server is seen in Figure. Four phases may be seen in the trade.

Phase 1. Establish Security Capabilities

Initiating a logical connection and determining the accompanying security capabilities are done during this step. The client starts the conversation by sending the message **client_hello** with the following parameters:

- **Version**
- **Random**
- **Session ID**
- **Cipher Suite**
- **Compression Method**

Phase 2. Server Authentication and Key Exchange

The initial step in this phase is initiated by the server, which sends its certificate, if authentication is necessary. The message comprises either a single X.509 certificate or a chain of such certificates. The certificate message is a mandatory component for any agreed-upon key exchange method, excluding

anonymous Diffie-Hellman. It is important to note that when fixed Diffie-Hellman is employed, this certificate message serves as the server's key exchange message, as it contains the server's public Diffie-Hellman parameters.

If necessary, a server_key_exchange message may be issued. In two situations, either (1) the server has sent a certificate with fixed Diffie-Hellman parameters or (2) an RSA key exchange is being utilized, it is not necessary.

The second parameter in the certificate_request message is a list of the distinguished names of acceptable certificate authorities.

Phase 3. Client authentication and key exchange

After receiving the server_done message, the client has certain responsibilities to fulfill. Firstly, it should verify the validity of the server's certificate (if it was required) and ensure that the server_hello parameters meet the acceptable criteria. If everything is in order, the client proceeds to send one or more messages back to the server. If the server has requested a certificate, the client initiates this phase by sending a certificate message. However, if a suitable certificate is not available, the client instead sends a no_certificate alert. Following that, the client must send the client_key_exchange message, which is a mandatory step in this phase.

Phase 4. Finish

This phase marks the conclusion of establishing a secure connection. The client initiates by transmitting a change_cipher_spec message and copying the pending CipherSpec into the current CipherSpec. It's worth noting that this message is not considered a component of the Handshake Protocol but is instead sent using the Change Cipher Spec Protocol. Subsequently, the client promptly sends the finished message using the new algorithms, keys, and secrets. The purpose of the finished message is to validate the success of the key exchange and authentication processes.

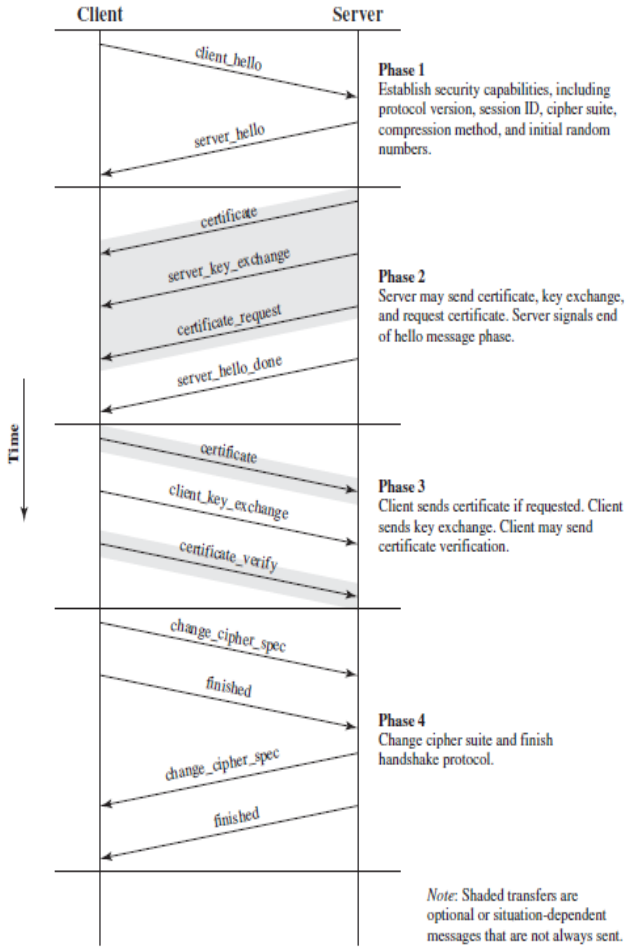

Fig: 3.8 Handshake Protocol Action

3.3 Transport Layer Security, HTTPS.

3.3.1 Transport Layer Security

TLS is an initiative by the Internet Engineering Task Force (IETF) aimed at creating an Internet standard edition of SSL. It

is formally defined as a Proposed Internet Standard in RFC 5246. This RFC, which outlines the specifications of TLS, shares many similarities with SSLv3. However, in this section, we emphasize the distinctions between the two protocols.

Version Number

The TLS Record Format is identical to the SSL Record Format, and the header fields have the same significance. The only variation lies in the version values. In the present TLS version, the major version is 3, and the minor version is also 3.c

Message Authentication Code

There are two distinctions between the SSLv3 and TLS MAC schemes: the specific algorithm used and the extent of the MAC calculation. TLS employs the HMAC (Hash-based Message Authentication Code) algorithm as defined in RFC 2104. As a reminder from Chapter 12, HMAC is characterized as:

$$\text{HMAC}_K(M) = \text{H}[(K^+ \oplus \text{opad}) \| \text{H}[(K^+ \oplus \text{ipad}) \| M]]$$

where

H	=	embedded hash function (for TLS, either MD5 or SHA-1)
M	=	message input to HMAC
K^+	=	secret key padded with zeros on the left so that the result is equal to the block length of the hash code (for MD5 and SHA-1, block length = 512 bits)
ipad	=	00110110 (36 in hexadecimal) repeated 64 times (512 bits)
opad	=	01011100 (5C in hexadecimal) repeated 64 times (512 bits)

Pseudorandom Function

For the purpose of key creation or validation, TLS uses a pseudorandom function called the PRF to expand secrets into

blocks of data. The goal is to produce larger blocks of data while using a shared secret value that is relatively modest and safe from attacks on MACs and hash algorithms. The data expansion function provides the foundation for the PRF.

```
P_hash(secret, seed) = HMAC_hash(secret,A(1) || seed) ||
                       HMAC_hash(secret, A(2) || seed) ||
                       HMAC_hash(secret, A(3) || seed) || . . .
```

where A() is defined as

A(0) = seed

A(i) = HMAC_hash(secret, A(i − 1))

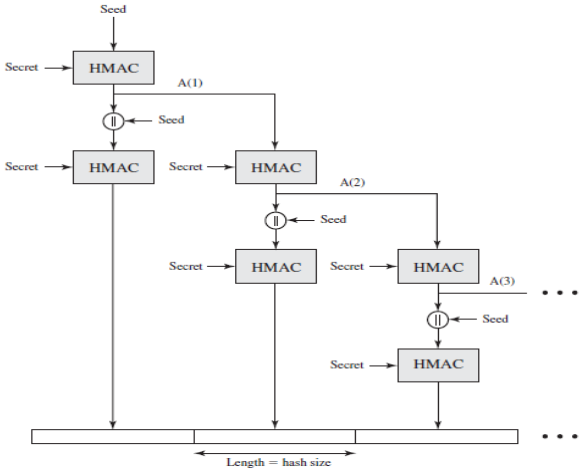

Figure 3.9 TLS Hash Function (Copy right is reserved to William Stallings.)

Alert Codes

TLS includes support for all the alert codes specified in SSLv3, except for "no_certificate." In addition to these, TLS introduces several new alert codes. Among the new codes, the following ones are always considered fatal.

- **record_overflow**
- **unknown_ca**
- **access_denied**
- **decode_error**
- **protocol_version**
- **insufficient_security**
- **unsupported_extension**
- **internal_error**
- **decrypt_error**

Cipher Suites

There are a few minor distinctions between the cipher suites offered in SSLv3 and those available in TLS:

- Key Exchange: TLS supports all key exchange techniques used in SSLv3, except for Fortezza.
- Symmetric Encryption Algorithms: TLS includes all symmetric encryption algorithms present in SSLv3, excluding Fortezza.

3.3.2 HTTPS

HTTPS (HTTP Secure) is a method of implementing secure communication between a web browser and a web server by combining HTTP with SSL (Secure Socket Layer) or its successor TLS (Transport Layer Security). Modern web browsers have built-in support for HTTPS, but its usage depends on the web server's support for HTTPS communication. It's important to note that some search engines may not support HTTPS.

The main difference noticeable to a web browser user is that URLs (Uniform Resource Locators) for secure connections start with "https://" instead of "http://". Regular HTTP connections typically use port 80, whereas HTTPS connections use port 443, which activates SSL or TLS for secure communication.

When utilizing HTTPS, the following components of the communication are encrypted for enhanced security:

- URL of the requested document.
- Contents of the document.
- Contents of browser forms filled in by the user.
- Cookies transmitted between the browser and the server.
- Contents of the HTTP header.

The specifications for HTTPS can be found in RFC 2818, which is titled "HTTP Over TLS." It's important to note that there are no substantial differences when using HTTP over either SSL or TLS, and both implementations are commonly referred to as HTTPS.

Connection Initiation

In the context of HTTPS, the HTTP client and TLS client are typically implemented together within the same software agent. This agent is responsible for initiating the secure connection with the server and performing the necessary TLS handshake.

To establish a connection, the HTTP client (which also acts as the TLS client) first initiates a TCP connection to the server on the appropriate port (usually port 443 for HTTPS). Once the

TCP connection is established, the TLS handshake process begins.

During the TLS handshake, the TLS client (i.e., the HTTP client) sends a TLS ClientHello message to the server. This message includes information such as the supported TLS versions, cipher suites, and any client-specific data required for the handshake. The server responds with a TLS ServerHello message, and the handshake continues with the exchange of additional messages, including the server's digital certificate (if required) and the generation of shared encryption keys.

Once the TLS handshake is successfully completed and the TLS session is established between the client and server, the HTTP client can proceed to initiate the first HTTP request. This request, along with any subsequent HTTP data, is encapsulated within TLS application data. The TLS layer encrypts the HTTP data before sending it over the established secure connection.

It's important to note that at the HTTP level, the client is typically unaware of the underlying transport layer (TCP or TLS). The HTTP client simply sends a connection request to the next lowest layer, which could be TCP or TLS. The establishment of a TCP connection is often involved in the process of setting up the TLS connection, but it's handled transparently by the TLS client implementation.

At the TLS level, a session is established between the TLS client and TLS server. This session can support multiple

connections simultaneously. This means that once the TLS session is established, the client can initiate multiple HTTP requests within that session without repeating the TLS handshake for each request. This allows for the efficient reuse of the established secure connection for subsequent HTTP requests, commonly known as connection pooling or connection reuse.

In summary, the HTTP client initiates a TCP or TLS connection to the server, performs the TLS handshake to establish a secure session, and then sends HTTP requests encapsulated within TLS application data over the secure connection. The TLS layer takes care of encrypting and decrypting the HTTP data, providing confidentiality and integrity for communication.

Connection Closure

In HTTPS, the closure of a connection can occur in different ways, depending on the circumstances and the intent of the client or server. Here are the common scenarios for HTTPS connection closure:

Normal closure: Either the client or the server can initiate a graceful closure of the connection after all the necessary data has been transmitted. This typically involves the client sending an HTTP request, the server responding with an HTTP response, and then both parties agreeing to close the connection. The closure process involves sending a TCP FIN (Finish) packet to signal the end of the data transmission. Once both sides acknowledge the FIN packet, the connection is closed.

Connection reuse: In HTTP/1.1, connections are often kept open for reuse. After a request-response cycle, the client and server can agree to reuse the existing connection for subsequent requests. In this case, the connection remains open, and further HTTP requests can be sent without the need for a complete reestablishment of the TLS session.

Connection timeout: If no activity occurs on the connection for a specified period, either the client or the server may choose to close the connection due to inactivity. This is known as a connection timeout. The timeout duration can vary depending on the implementation or configuration.

Error or exception: In case of errors or exceptions during the connection or data transmission process, either the client or the server may choose to terminate the connection abruptly. This can happen if there is a protocol violation, a TLS handshake failure, a network issue, or any other situation that prevents the normal functioning of the connection.

Client-initiated closure: The client can explicitly request the closure of the connection if it has finished sending requests and does not require further communication with the server. This can be done by including the "Connection: close" header in the HTTP request, indicating to the server that the client wants the connection to be closed after the response.

Server-initiated closure: Similarly, the server can also explicitly request the closure of the connection by including the "Connection: close" header in the HTTP response. This signals to the client that the server intends to close the connection after sending the response.

It's worth noting that in all these scenarios, the closure of the HTTPS connection follows the same underlying principles of closing the TCP connection, as HTTPS operates over TCP. The TLS layer, which sits on top of TCP, handles the secure transmission of the HTTP data. When the HTTPS connection is closed, it involves closing the TLS session, which in turn closes the underlying TCP connection.

3.4 Secure Shell (SSH)T1

3.4.1 Secure Shell

SSH, also known as Secure Shell, is a communication protocol designed to ensure secure network connections. It was created with the intention of being relatively straightforward and cost-effective to implement. The initial release, SSH1, aimed to provide a secure method for remote logins, replacing insecure options like TELNET and other similar logon systems. Additionally, SSH offers broader client/server capabilities and can be employed for various network tasks like file transfer and email. To address security vulnerabilities present in the original version, SSH2 was developed, and it rectifies a range of flaws. The specifications for SSH2, outlined in IETF RFCs 4250 through 4256, are documented as a proposed standard.

SSH is organized as three protocols that typically run on top of TCP (Figure 3.10):

1. Transport Layer Protocol: Provides server authentication, data confidentiality, and data integrity with forward secrecy (i.e., if a key is compromised during one session, the knowledge does not affect the security of earlier sessions). The transport layer may optionally provide compression.

2. User Authentication Protocol: Authenticates the user to the server.

3. Connection Protocol: Multiplexes multiple logical communications channels over a single, underlying SSH connection

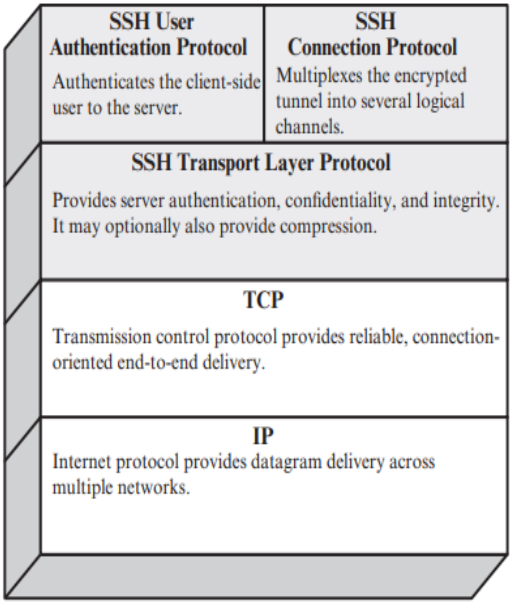

Figure 3.10 SSH Protocol Stack (Copy right is reserved to William Stallings.)

3.4.2 Transport Layer Protocol

The authentication of servers in SSH takes place at the transport layer, relying on the server's possession of a public/private key pair. A server can have multiple host keys implemented using various asymmetric encryption algorithms. It is also possible for multiple hosts to share the same host key.

During the key exchange process, the server host key is utilized to verify the identity of the host. However, for this verification to occur, the client must possess prior knowledge of the server's public host key. RFC 4251 defines two alternative trust models that can be employed in this context.

a. The client maintains a local database that connects each user-typed host name with its corresponding public host key. This approach eliminates the need for any centralized administration or third-party coordination. However, a potential drawback is that the database containing the associations between names and keys may become challenging to manage and maintain over time.
b. The association between host names and keys is validated by a trusted certification authority (CA). The client is aware of the CA root key and can authenticate the legitimacy of all host keys certified by recognized CAs. This alternative approach simplifies the maintenance concern as only a single CA key needs to be securely stored on the client. However, it necessitates that each host key is duly certified by a central authority before authorization can take place.

Packet Exchange

To begin with, the client initiates a TCP connection with the server. This connection is established using the TCP protocol and occurs separately from the Transport Layer Protocol. Once the connection is successfully made, the client and server engage

in the exchange of data packets within the data field of a TCP segment. The following are the packet format:

- Packet length
- Padding length
- Payload
- Random padding
- Message authentication code (MAC)

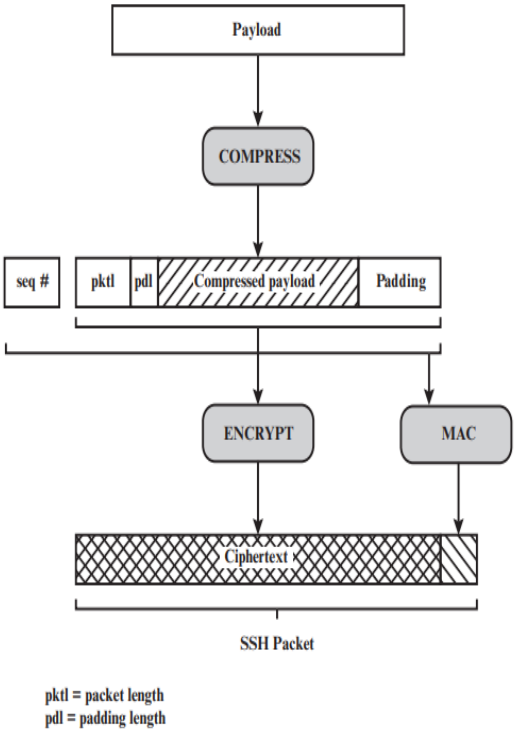

Figure 3.11: SSH Transport Layer Protocol Packet Formation. (Copy right is reserved to William Stallings.)

Key Generation

The encryption and MAC keys, along with any necessary initialization vectors (IVs), are derived from the shared secret key K, the hash value from the key exchange H, and the session identifier. The session identifier is typically equal to H, unless there has been an additional key exchange following the initial one. The computation of these values is as follows:

- Initial IV client to server: HASH($K \| H \| \text{“A”} \| \text{session\_id}$)
- Initial IV server to client: HASH($K \| H \| \text{“B”} \| \text{session\_id}$)
- Encryption key client to server: HASH($K \| H \| \text{“C”} \| \text{session\_id}$)
- Encryption key server to client: HASH($K \| H \| \text{“D”} \| \text{session\_id}$)
- Integrity key client to server: HASH($K \| H \| \text{“E”} \| \text{session\_id}$)
- Integrity key server to client: HASH($K \| H \| \text{“F”} \| \text{session\_id}$)

3.4.3 User Authentication Protocol

The User Authentication Protocol facilitates the process of authenticating the client to the server.

Message Types and Formats

Three types of messages are always used in the User Authentication Protocol. Authentication requests from the client have the format:

byte	SSH_MSG_USERAUTH_REQUEST (50)
string	user name
string	service name
string	method name
...	method specific fields

Message Exchange

The message exchange involves the following steps:

1.The client transmits a SSH_MSG_USERAUTH_REQUEST message with the requested authentication method set to "none".

2.The server verifies the validity of the username. If the username is invalid, the server responds with SSH_MSG_USERAUTH_FAILURE and the partial success value set to false. If the username is valid, the server proceeds to the next step.

3.The server sends SSH_MSG_USERAUTH_FAILURE message, containing a list of one or more authentication methods to be utilized.

4.The client selects one of the acceptable authentication methods from the list provided and sends a SSH_MSG_USERAUTH_REQUEST message with the chosen method name and any required method-specific fields. This step may involve a sequence of exchanges to complete the chosen authentication method.

5. If the authentication is successful and additional authentication methods are needed, the server advances to step 3, indicating a partial success value of true. In the case of authentication failure, the server also proceeds to step 3 but with a partial success value of false.

6. Once all the required authentication methods have successfully completed, the server transmits a

SSH_MSG_USERAUTH_SUCCESS message, marking the conclusion of the Authentication Protocol.

Authentication Methods

The server may require one or more of the following authentication methods:

- Publickey
- password
- hostbased

3.4.4 Connection Protocol

Channel Mechanism

The SSH Connection Protocol operates on the SSH Transport Layer Protocol and assumes the presence of a secure authentication connection. This secure authentication connection, known as a tunnel, is utilized by the Connection Protocol to multiplex several logical channels.

SSH supports various forms of communication, including terminal sessions, by employing distinct channels. Both the client and server have the ability to initiate a channel. Each channel is assigned a unique channel number, which may differ between the client and server. Flow control for channels is achieved through a window mechanism. It is important to note

that data cannot be transmitted to a channel until a message is received, indicating the availability of window space.

When either the client or server intends to establish a new channel, it assigns a local number to the channel and subsequently transmits a message in the following format:

byte	SSH_MSG_CHANNEL_OPEN
string	channel type
uint32	sender channel
uint32	initial window size
uint32	maximum packet size
....	channel type specific data follows

Once a channel has been successfully opened, the transfer of data occurs through the utilization of SSH_MSG_CHANNEL_DATA messages. These messages contain both the recipient channel number and a block of data to be transmitted. The exchange of such messages can persist in both directions as long as the channel remains open.

When either the client or server intends to terminate a channel, it sends a SSH_MSG_CHANNEL_CLOSE message. This message includes the recipient channel number, indicating the channel to be closed.

Channel Types

Four channel types are recognized in the SSH Connection Protocol specification:

- Session
- x11

- forwarded-tcpip
- direct-tcpip

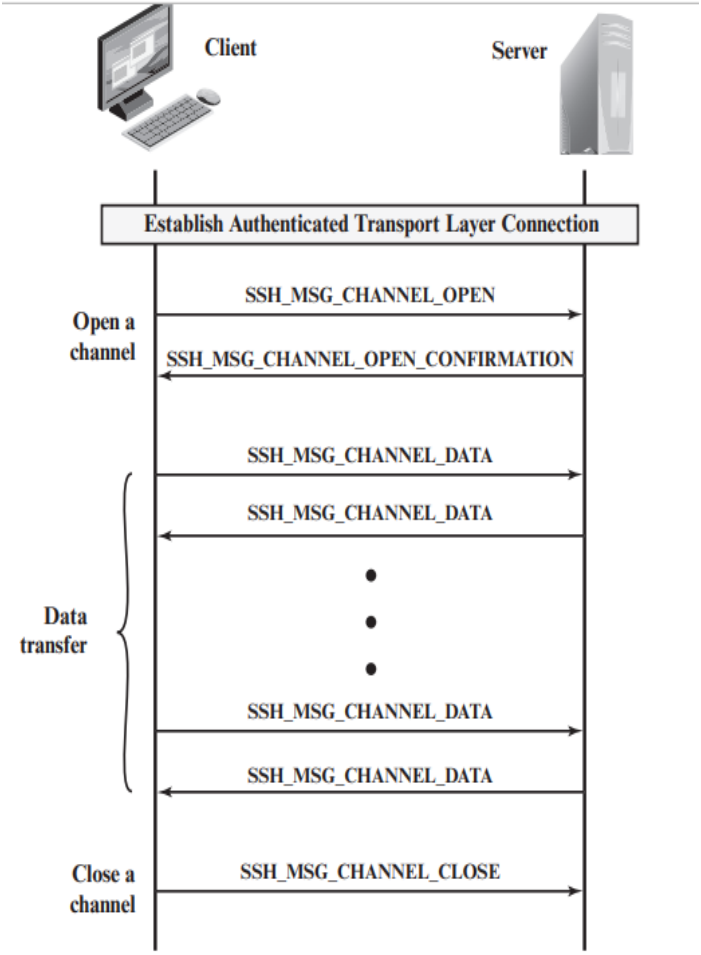

Figure 3.12: Example of SSH Connection Protocol Message Exchange. (Copy right is reserved to William Stallings.)

Port Forwarding

Port forwarding is a highly valuable feature provided by SSH, allowing the conversion of any insecure TCP connection into a secure SSH connection. This functionality is also commonly known as SSH tunneling. To comprehend the concept of port forwarding, it is essential to understand the definition of a port within this context. In TCP, a port serves as an identifier for a specific user or application. Every application that operates over TCP is assigned a port number. Incoming TCP traffic is directed to the appropriate application based on the designated port number. It is possible for an application to utilize multiple port numbers.

For instance, let's consider the Simple Mail Transfer Protocol (SMTP). The server side of SMTP typically listens on port 25. When an incoming TCP connection contains SMTP data and is addressed to destination port 25, TCP recognizes that this data is intended for the SMTP server application and delivers it accordingly.

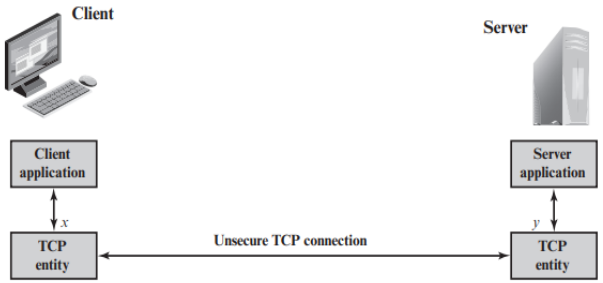

Figure 3.13: Connection via TCP. (Copy right is reserved to William Stallings.)

Within our scenario, there is a client application identified by a specific port number, denoted as x, and a server application identified by another port number, referred to as y. At a certain stage, the client application initiates a request to establish a connection with the remote server via the local TCP entity, specifically targeting port y. The local TCP entity proceeds to negotiate a TCP connection with the corresponding remote TCP entity, resulting in the establishment of a connection that links the local port x to the remote port y.

To ensure the security of this connection, SSH is configured in a way that the SSH Transport Layer Protocol establishes a TCP connection between the SSH client and server entities. The client entity utilizes TCP port number a, while the server entity uses TCP port number b. Over this established TCP connection, a secure SSH tunnel is formed. Consequently, any traffic originating from the client application on port x is redirected to the local SSH entity. This traffic then travels through the SSH tunnel, where the remote SSH entity receives and delivers the data to the server application on port y. Similarly, traffic in the opposite direction undergoes a similar redirection process.

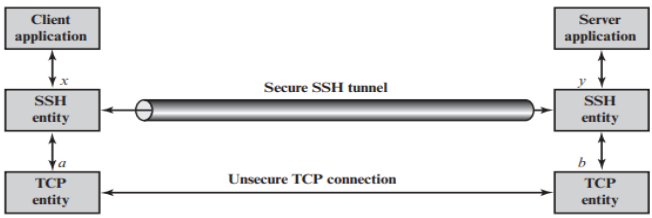

Figure 3.14: Connection via SSH (Copy right is reserved to William Stallings.)

SSH offers two types of port forwarding: local forwarding and remote forwarding. Local forwarding enables the client to establish a "hijacker" process that intercepts specific application-level traffic and redirects it from an insecure TCP connection to a secure SSH tunnel. The SSH configuration includes designated ports on which SSH listens. Any traffic received on these selected ports is captured by SSH and transmitted through the SSH tunnel. At the other end of the tunnel, the SSH server receives the incoming traffic and forwards it to the destination port specified by the client application.

In the case of remote forwarding, the SSH client operates on behalf of the server. The client receives incoming traffic destined for a specific port number, directs the traffic to the appropriate port, and forwards it to the destination specified by the user. A common example of remote forwarding is when you want to access a server at your workplace from your home computer. Due to the presence of a firewall, the work server does not accept SSH requests from external devices like your home computer. However, from your workplace, you can establish an SSH tunnel using remote forwarding, allowing you to securely access the work server from your home computer.

3.5 Secure electronic transaction (SET)

3.5.1 SET Process

Secure Electronic Transaction (SET) was a protocol developed in the 1990s as a security framework for online credit card transactions. It was designed to ensure the confidentiality, integrity, and authentication of electronic payment information.

The SET protocol aimed to provide a secure and trustworthy environment for conducting e-commerce transactions over the Internet. It involved the collaboration of payment card networks, merchants, cardholders, and certificate authorities to establish a secure infrastructure.

Here's a brief overview of how SET worked:

a. Registration: Cardholders and merchants had to register with a trusted certificate authority to obtain digital certificates.
b. Digital Certificates: Digital certificates were used to verify the identities of participants in a transaction. They contained public keys, which were used for encryption and authentication.
c. Secure Channel: SET established a secure channel between the cardholder and the merchant, encrypting the data exchanged during the transaction.

d. Payment Authorization: The cardholder's payment information was encrypted and sent to the merchant, who forwarded it to the cardholder's payment gateway.
e. Dual Encryption: SET utilized dual encryption, where the cardholder encrypted payment information with the merchant's public key, and the merchant encrypted the payment authorization request with the payment gateway's public key.
f. Payment Processing: The payment gateway decrypted the request, verified the digital certificates, and forwarded the payment information to the appropriate card network for authorization.
g. Authorization and Settlement: The card network received the request, verified the transaction details, and communicated with the issuing bank to authorize or decline the transaction. If authorized, the settlement process was initiated.

While SET was developed with good intentions, it faced challenges in adoption due to its complexity, implementation costs, and the rise of alternative payment solutions. As a result, other secure payment protocols and technologies, such as SSL/TLS and 3D Secure, gained more popularity in securing online transactions.

Examples of SET

To better understand SET and its role in electronic transactions, let's consider a typical scenario involving various participants: a client, a payment gateway, a client's financial institution, a merchant, and a merchant's financial institution.

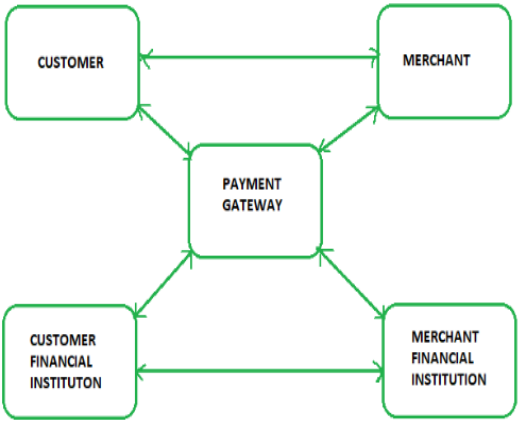

Figure3.15: SET Example. (Copy right is reserved to William Stallings.)

3.5.2 History of SET

In response to the rise of e-commerce transactions, particularly consumer-driven online purchases, secure electronic transaction protocols were developed. During the mid-1990s, conducting business on the internet was a novel concept, and the available security measures for protecting these transactions were still evolving and varied in their effectiveness.

To enable the use of online payment systems by retailers and financial institutions, it was necessary to establish protocols that provided the required security. The secure electronic transaction standards defined these protocols, ensuring that entities possessing the appropriate software could decrypt and process digital transactions accurately.

In 1996, the SET Consortium, a collaborative group comprising VISA, Mastercard, GTE, IBM, Microsoft, Netscape, SAIC, Terisa Systems, RSA, and VeriSign, set a common objective. They aimed to unify incompatible security protocols, namely STT from Visa and Microsoft, and SEPP from Mastercard and IBM, into a single comprehensive standard. This effort aimed to create a cohesive and widely accepted framework for secure electronic transactions.

3.5.3 Requirements in SET

In the SET protocol, several key requirements were established to ensure secure electronic transactions. These requirements included:

a. Confidentiality: SET aimed to maintain the confidentiality of sensitive payment information during transmission. This involved encrypting the data exchanged between the client, payment gateway, and other involved parties, preventing unauthorized access.

b. Integrity: SET ensured the integrity of transaction data by utilizing cryptographic techniques. This involved using digital signatures to verify the authenticity of messages and detect any tampering or modifications during transmission.
c. Authentication: SET focused on authenticating the identities of the participating entities involved in the transaction. This was achieved through the use of digital certificates issued by trusted certificate authorities, which validated the legitimacy of the involved parties.
d. Non-repudiation: SET aimed to provide non-repudiation, meaning that once a transaction was conducted, the parties involved could not deny their participation. Digital signatures were used to create evidence of the transaction, ensuring accountability, and preventing disputes.
e. Compatibility: SET was designed to be compatible with existing financial networks, payment systems, and infrastructures. It aimed to integrate seamlessly with established protocols and standards to facilitate widespread adoption.
f. Payment Processing: SET included mechanisms for secure payment processing. This involved the encryption and transmission of payment information from the client to the merchant, through the payment gateway, and

ultimately to the client's financial institution for authorization and settlement.

By addressing these requirements, SET aimed to establish a secure framework for electronic transactions, providing confidence to both clients and merchants in conducting online payments.

3.5.4 Participants in SET

In the general scenario of online transactions, SET includes similar participants:

1. Cardholder – customer
2. Issuer – customer financial institution
3. Merchant
4. Acquirer – Merchant financial
5. Certificate authority – Authority that follows certain standards and issues certificates (like X.509V3) to all other participants.

3.5.5 Working Principle in SET

Before engaging in online transactions, both cardholders and merchants are required to register with a certificate authority (CA). Once the registration process is completed, cardholders and merchants can initiate transactions using a simplified protocol consisting of nine basic steps:

- The customer explores the website and selects the desired items for purchase.

- The customer sends an order along with payment information, combining two components within a single message:
    1. The purchase order, intended for the merchant.
    2. The card information, exclusively for the merchant's bank.
- The merchant forwards the card information to their bank.
- The merchant's bank verifies the payment authorization by contacting the card issuer.
- The card issuer sends the authorization response to the merchant's bank.
- The merchant's bank relays the authorization to the merchant.
- The merchant completes the order and sends a confirmation to the customer.
- The merchant captures the transaction from their bank.
- The card issuer generates a credit card bill (invoice) for the customer.

These steps outline a simplified process, allowing cardholders and merchants to conduct secure online transactions while ensuring the flow of information and authorization between the involved parties.

3.5.6 SET functionalities:

a. Authentication:

Merchant Authentication: SET incorporates the use of X.509V3 certificates to enable customers to authenticate merchants and verify their previous relationships with financial institutions. This measure helps prevent theft and ensure trustworthiness in transactions.

Customer/Cardholder Authentication: SET employs X.509V3 certificates to verify the authorization of credit card usage by customers, ensuring that only authorized users can make transactions.

b. Message Confidentiality:

SET ensures message confidentiality by utilizing encryption techniques. The protocol traditionally employs DES (Data Encryption Standard) for encryption purposes, safeguarding the transferred message from unauthorized access and ensuring its privacy.

c. Message Integrity:

SET protects messages from unauthorized modification by implementing digital signatures. RSA digital signatures with SHA-1 (Secure Hash Algorithm 1) are commonly used to ensure message integrity. Additionally, some messages employ HMAC (Hash-based Message Authentication Code) with SHA-1 for added protection against unauthorized modifications. These

measures ensure the integrity and authenticity of the transmitted data.

3.5.7 SET – Payment Processing

Dual Signature

The concept of dual signature was introduced in the SET protocol to establish a connection between two distinct information components intended for two separate recipients:

1. Order Information (OI) for merchant
2. Payment Information (PI) for bank

While sending the two information pieces separately may seem like an easy and potentially more secure approach, combining them in a connected form through dual signature generation resolves any potential future disputes. Here is the process of generating the dual signature:

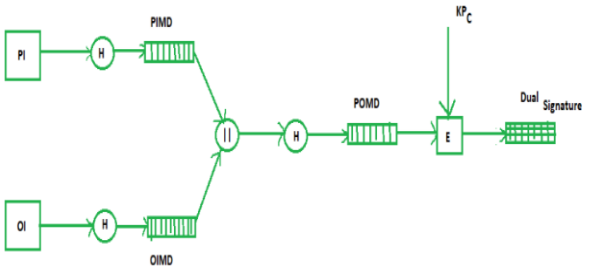

Figure 3.16: Dual Signature Generation (Copy right is reserved to William Stallings.)

Where

- PI stands for payment information,
- OI stands for order information,
- PIMD stands for Payment Information Message Digest,
- OIMD stands for Order Information Message Digest,
- POMD stands for Payment Order Message Digest,
- H stands for Hashing,
- E stands for public key encryption,
- KPc is customer's private key,
- || stands for append operation,
- Dual signature, DS= E(KPc, [H(H(PI)||H(OI))]),

To generate a purchase request, the process involves three essential inputs:

a. Payment Information (PI): This includes the necessary details related to the payment, such as credit card information, billing address, and transaction amount.
b. Dual Signature: The dual signature, specific to SET, is a combination of two separate signatures. It connects the payment information and the order information message digest, ensuring their association and integrity.
c. Order Information Message Digest (OIMD): The OIMD represents a condensed and unique representation of the order information. It is typically generated using a

cryptographic hash function, such as SHA-1 or SHA-256, to produce a fixed-size digest.

Together, these inputs form the basis for generating a purchase request in the SET protocol, facilitating secure and interconnected information exchange between the involved parties.

The steps involved in generation of Purchase information is as follows:

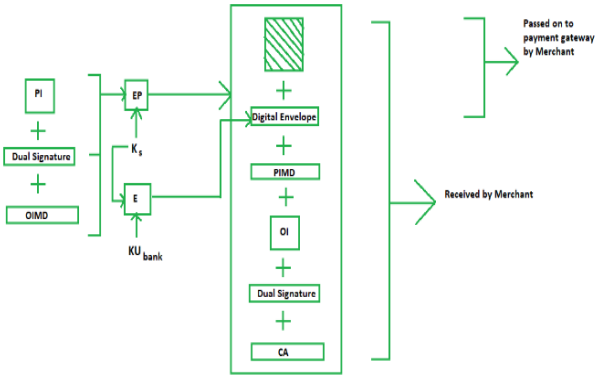

Figure 3.17: Purchase Information Generation. (Copy right is reserved to William Stallings.)

Here,

PI, OIMD, OI all are with the same meanings as before.

The new things are:

EP which is symmetric key encryption.

Ks is a temporary symmetric key.

KUbank is public key of bank.

CA is Cardholder or customer Certificate.

Digital Envelope = E(KUbank, Ks).

3.5.8 Example Of SET

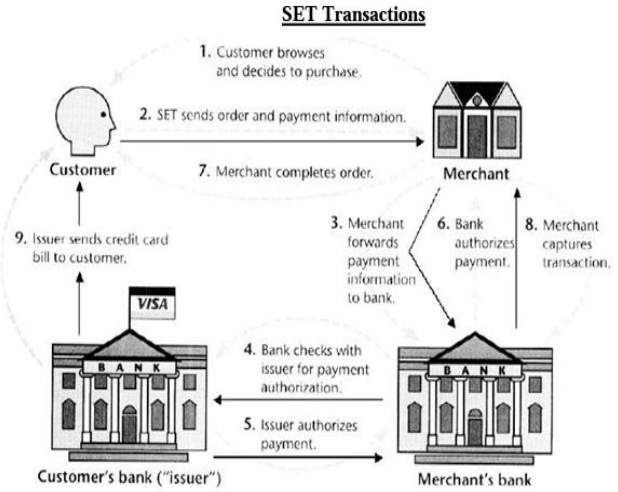

Figure 3.18: Electronic Transaction. (Copy right is reserved to William Stallings.)

Purchase Request Validation on Merchant Side: The Merchant verifies by comparing POMD generated through PIMD hashing with POMD generated through decryption of Dual Signature as follows:

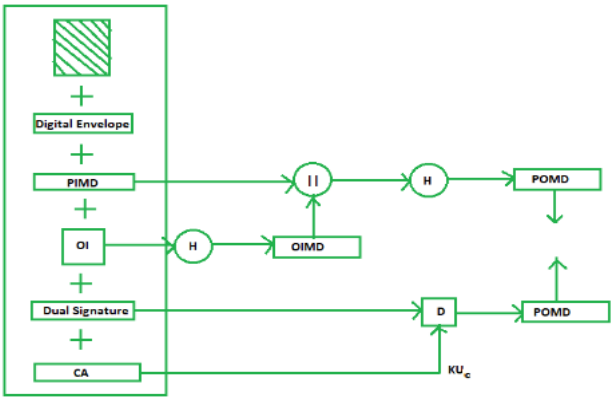

Figure 3.19 Dual Signature with POMD. (Copy right is reserved to William Stallings.)

Payment Authorization: This step involves the merchant's authorization of the payment information provided by the customer. It ensures that the payment will be received by the merchant. The authorization process typically involves verifying the payment details, checking for available funds, and confirming the transaction's legitimacy. Once authorized, the merchant can proceed with fulfilling the order.

Payment Capture: Payment capture refers to the process of the merchant receiving the actual payment from the customer. After the authorization is obtained, the merchant generates request blocks containing the necessary information and sends them to the payment gateway. The payment gateway, acting as an

intermediary, processes the payment and initiates the transfer of funds from the customer's account to the merchant's account. This completes the payment capture process, allowing the merchant to successfully receive the payment for the goods or services provided.

These steps, involving authorization and capture, ensure a secure and reliable flow of funds in the electronic transaction, providing confidence to both the merchant and the customer.

3.6 Security vulnerabilities – Scanning techniques, Vulnerability assessment - Penetration testing.

3.6.1 Security Vulnerabilities –Scanning Techniques.

Security vulnerabilities can be exploited through various scanning techniques that are used by attackers to identify weaknesses in systems or networks. These scanning techniques include:

a. **Port Scanning**: Attackers use port scanning to discover open ports on a target system. By scanning different ports, they can identify potential entry points and vulnerabilities that can be exploited.

b. **Network Scanning**: Network scanning involves mapping the network infrastructure to identify active hosts, services, and devices. Attackers use network scanning to gather

information about the network architecture and potential vulnerabilities.

c. **Vulnerability Scanning:** Vulnerability scanning tools are used to identify known vulnerabilities in software, systems, or network components. These tools scan for specific weaknesses, such as outdated software versions, misconfigurations, or known security flaws.
d. **Web Application Scanning:** Web application scanning focuses on identifying vulnerabilities in web applications. Attackers can use automated tools to scan web applications for common weaknesses like SQL injection, cross-site scripting (XSS), or insecure configurations.
e. **Wireless Network Scanning:** Attackers can scan wireless networks to discover and exploit vulnerabilities in wireless protocols, weak encryption methods, or insecure network configurations.
f. **Host Scanning:** Host scanning involves examining specific hosts or systems to identify potential vulnerabilities. Attackers can scan for open services, misconfigured security settings, or weak authentication mechanisms.

It's important to note that while these scanning techniques can be used by attackers, they can also be employed by security professionals to assess and mitigate vulnerabilities proactively.

Regular security assessments and scanning can help identify and address weaknesses before they are exploited.

Vulnerability scanning plays a crucial role in enabling organizations to examine potential threats across their complete IT infrastructures, encompassing software, specialized devices, files, and databases. While the specific types of vulnerability scanning may differ, the underlying technique remains an integral part of any comprehensive cybersecurity management program.

By conducting vulnerability scanning, organizations can identify weaknesses and potential points of risk. This proactive approach allows for the prompt remediation of vulnerabilities before they have the chance to become disruptive. Ultimately, vulnerability scanning helps ensure the security and stability of the IT environment, safeguarding against potential exploits and minimizing the potential impact of security breaches.

3.6.2 Working of Vulnerability Scanning

Vulnerability scanning plays a crucial role in enabling organizations to examine potential threats across their complete IT infrastructures, encompassing software, specialized devices, files, and databases. While the specific types of vulnerability scanning may differ, the underlying technique remains an integral part of any comprehensive cybersecurity management program.

By conducting vulnerability scanning, organizations can identify weaknesses and potential points of risk. This proactive approach allows for the prompt remediation of vulnerabilities before they have the chance to become disruptive. Ultimately, vulnerability scanning helps ensure the security and stability of the IT environment, safeguarding against potential exploits and minimizing the potential impact of security breaches.

Vulnerability scanning is commonly carried out using dedicated software designed to analyze network-connected assets. This software leverages a database containing known anomalies, often sourced from the Common Vulnerabilities and Exposures (CVE) database, which catalogues publicly disclosed threats. When the scanning software detects any deviations from the expected norms, it generates an alert that is displayed on the user's dashboard. This enables prompt identification of potential vulnerabilities and facilitates proactive measures to mitigate risks.

Vulnerability scanners fulfil three primary functions:

a. **Discovery:** Scanning enables the identification of both known and previously unknown assets, providing a snapshot or continuous view of the network at a specific point in time.
b. **Assessment:** These systems detect and highlight vulnerabilities by utilizing data from the Common Vulnerabilities and Exposures (CVE) database and other

repositories of known anomalies. This assessment helps in identifying potential security weaknesses within the network infrastructure.

c. **Prioritization:** Using programmed metrics, the vulnerability scanner leverages data from the CVE database, threat intelligence, and data science to assess the severity of vulnerabilities and prioritize them for remediation based on their potential impact.

It is important to differentiate vulnerability scanning from penetration testing. While vulnerability scanning identifies issues, penetration testing goes a step further by exploring anomalies in detail and providing methods to address and eliminate them. Together, these systems create a powerful combination that equips cybersecurity managers and technical staff with comprehensive tools to enhance security and effectively manage potential risks.

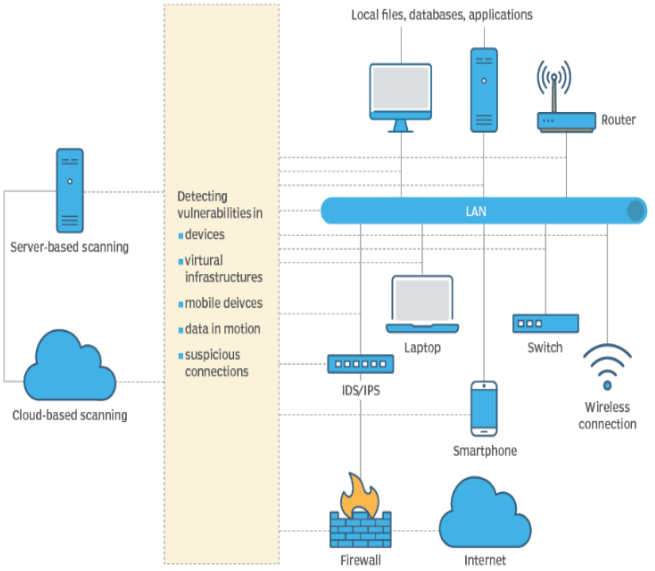

Figure 3.20: Vulnerability Scanning working. (Copy right is reserved to William Stallings.)

3.6.3 Types of Vulnerability Scanning and Applications

The capabilities of vulnerability scanners can vary, and they can be classified into different types, including:

a. Internal Scanning: This type of scanning occurs within the network boundaries and focuses on strengthening applications and resources against internal threats, such as rogue employees or attackers who have breached the network perimeter.

b. External Scanning: Operating from an external perspective, this scanning method simulates the actions of a threat actor to identify vulnerabilities within the organization's

perimeter. It targets components like firewalls, internal applications, web apps, data ports (especially open and underutilized ones), and network elements.

c. Authenticated Scanning: With authenticated scanning, a tester logs in as a legitimate user and examines vulnerabilities from a trusted user's perspective, providing insights into potential risks from within the authenticated user's privileges.

d. Unauthenticated Scanning: Unauthenticated scanning involves an unauthenticated tester examining the infrastructure as an intruder, aiming to identify additional risks and vulnerabilities that may be accessible without authentication.

e. Assessment Scanning: This fundamental scanning activity provides an overview and analysis of the infrastructure, identifying potential weaknesses and vulnerabilities.

f. Discovery Scanning: Discovery scanning is a core function of any scanner, focused on identifying situations that pose risks and threats to the organization.

g. Compliance Scanning: This type of scanning determines whether the infrastructure adheres to standards, policies, regulations, and other rule-based requirements, ensuring compliance with relevant guidelines.

h. Host-based Scanning: Host-based scanners assess local systems, servers, and their operating system configurations,

along with other supported hosts, to identify vulnerabilities specific to those hosts.

i. Network Scanning: Network scanning works in conjunction with port scanners to check for weak or questionable passwords and perform limited penetration testing. It identifies vulnerabilities and attack vectors without disrupting network operations or affecting system performance.

j. Web Application Scanning: This type of scanning focuses on examining public-facing web applications to identify potential vulnerabilities that could be exploited by attackers.

k. Port Scanning: Port scanning involves searching for open ports on network servers by sending connection requests and monitoring the responses. It helps identify open or underutilized ports that could pose security risks.

l. Database Scanning: This technique involves probing databases to uncover any suspicious activity or vulnerabilities within the database infrastructure.

m. Source Code Vulnerability Scanning: Regular examination of source code helps testers identify potential anomalies and vulnerabilities. Vulnerability scanning software may utilize anomaly data gathered by organizations like NIST (National Institute of Standards and Technology).

n. Cloud Vulnerability Scanning: Cloud scanners focus on assessing the organization's cloud-based environment to

identify potential anomalies. They look for issues such as misconfigurations, access control problems, authentication weaknesses, and conflicts with other cloud users.

These different types of vulnerability scanning techniques provide organizations with a range of options to assess their systems, infrastructure, and applications for potential security risks and vulnerabilities.

3.6.4 Tools Used for Vulnerability Assessment

There are several tools available for conducting vulnerability assessments to identify and analyze potential security vulnerabilities in systems and networks. Some commonly used tools for vulnerability assessment include:

a. Nessus: Nessus is a widely recognized and popular vulnerability scanner that performs comprehensive assessments of network infrastructure and identifies potential vulnerabilities across various platforms and devices.
b. OpenVAS: OpenVAS (Open Vulnerability Assessment System) is an open-source vulnerability scanner that provides a framework for vulnerability scanning and management. It offers a wide range of scanning capabilities and is regularly updated with new vulnerability checks.
c. Qualys Vulnerability Management: Qualys offers a suite of vulnerability management tools that include scanning

capabilities, asset discovery, vulnerability prioritization, and reporting. It provides comprehensive vulnerability assessment and management solutions for organizations of all sizes.

d. Rapid7 Nexpose: Nexpose is a vulnerability management tool that offers comprehensive scanning and assessment capabilities. It helps identify vulnerabilities across a wide range of assets and provides detailed reports and remediation recommendations.

e. Nikto: Nikto is an open-source web server scanner that specializes in identifying vulnerabilities and misconfigurations in web servers. It scans for common web application vulnerabilities and provides detailed reports of its findings.

f. Burp Suite: Burp Suite is a popular web application testing tool that includes a vulnerability scanner. It offers a range of features, including web application scanning, manual penetration testing, and advanced vulnerability detection.

g. Acunetix: Acunetix is a web vulnerability scanner that helps identify security flaws in web applications, including SQL injection, cross-site scripting (XSS), and other common vulnerabilities. It provides detailed reports and prioritizes vulnerabilities based on severity.

h. OpenSCAP: OpenSCAP is an open-source security compliance assessment tool that supports vulnerability

scanning and configuration compliance checking. It provides detailed reports and can be integrated into existing security frameworks.

i. Retina: Retina is a vulnerability management tool that offers scanning capabilities for identifying vulnerabilities in networks, systems, and applications. It provides comprehensive reports and helps prioritize remediation efforts.

j. Microsoft Baseline Security Analyzer (MBSA): MBSA is a free Microsoft tool that scans Windows-based systems for common security misconfigurations and vulnerabilities. It helps assess the security posture of Windows-based environments.

These are just a few examples of the many tools available for vulnerability assessment. The choice of tool depends on the specific requirements of the organization and the nature of the systems and networks being assessed.

3.7 Terminal Questions:

1. Are you satisfied with the insights and recommendations provided regarding web security, or is there any further assistance you require?
2. Have you taken any steps towards implementing or enhancing web security measures based on our discussion today, or do you need further guidance on how to proceed?
3. How does SSL/TLS handle encryption key management?
4. Explain the concept of forward secrecy in SSL/TLS.

5. What is the significance of certificate revocation in SSL/TLS?
6. Describe the role of cipher suites in SSL/TLS.
7. How can you monitor and troubleshoot SSL/TLS connections?
8. Explain the role of digital certificates in TLS and how they contribute to establishing a secure connection.
9. Describe the TLS handshake process and the key steps involved.
10. How does TLS ensure the confidentiality and integrity of data transmission?
11. What role does Transport Layer Security (TLS) play in establishing a secure HTTPS connection?
12. How does HTTPS ensure the confidentiality of data transmitted between a client and a server?
13. Discuss the process of validating the authenticity of a website's SSL/TLS certificate in HTTPS.
14. How can you enable verbose output during an SSH connection for troubleshooting purposes?
15. How do you securely copy a directory from the local system to a remote server using SCP?
16. How do you securely transfer files using SFTP?
17. Explain the key security objectives of SET and how they are achieved.

18. Describe the cryptographic techniques used in SET to ensure secure transactions.
19. How does SET ensure the confidentiality and integrity of payment data during transmission?
20. Describe the step-by-step process of a typical SET transaction, from customer registration to payment settlement.
21. What is the impact of security vulnerabilities on systems and networks?
22. How can organizations identify and assess security vulnerabilities?
23. What is the importance of prioritizing vulnerabilities based on their severity?
24. What role does patch management play in addressing security vulnerabilities?
25. What are some best practices for preventing security vulnerabilities?

Phone 8870225085
Email: roshinicse22@kluniversity.in
https:kluniversity.in

Chapter Four - IP Security

4.1 Pretty Good Privacy, S/MIME

Basically**, Email security** refers to the steps we take to protect the email messages and the information that they contain from un authorized access and damage. It involves ensuring the confidentiality, integrity, and availability of email messages, as well as safeguarding against phishing attacks, spam, viruses, and another form of malware. It can be achieved through a combination of technical and non-technical measures.

Some standard technical measures include the encryption of email messages to protect their contents, the use of digital signatures to verify the authenticity of the sender, and email filtering systems to block unwanted emails and malware. Non-technical measures may include training employees on how to recognize and respond to phishing attacks and other email security threats, establishing policies and procedures for email use and management, and conducting regular security audits to identify and address vulnerabilities.

We can say that email security is important to protect sensitive information from unauthorized access and ensure the reliability and confidentiality of electronic communication.

4.1.1 Steps to Secure Email:

We can take the following actions to protect our email:

- Choose a secure password that is at least 12 characters long and contains uppercase and lowercase letters, digits, and special characters.

- Activate two-factor authentication, which adds an additional layer of security to your email account by requiring a code in addition to your password.
- Use encryption, which encrypts your email messages so that only the intended receiver can decipher them. Email encryption can be done by using programs like PGP or S/MIME.
- Keep your software up to date. Ensure that the most recent security updates are installed on your operating system and email client.

- Beware of phishing scams: Hackers try to steal your personal information by pretending to be someone else in phishing scams. Be careful of emails that request private information or have suspicious links because these are the resources of the phishing attack.
- Choose a trustworthy email service provider**.** Search for a service provider that protects your data using encryption and other security measures.

- Use a VPN: Using a VPN can help protect our email by encrypting our internet connection and disguising our IP address, making it more difficult for hackers to intercept our emails.
- Upgrade Your Application Regularly**:** People now frequently access their email accounts through apps,

although these tools are not perfect and can be taken advantage of by hackers. A cybercriminal might use a vulnerability, for example, to hack accounts, steal data, or send spam mail. Because of this, it's important to update your programs frequently.

4.1.2 Pretty Good Privacy (PGP):

PGP is a remarkable phenomenon. Largely the effort of a single person, Phil Zimmermann, PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. In essence, what Zimmermann has done is the following:

Selected the best cryptographic mechanisms (algorithms) as building blocks.

Integrated these algorithms into a general-purpose application that is independent of the operating system and processor and that is based on a small set of easy-to-use commands.

Made the package and its source code freely available via the Internet, bulletin boards, and commercial networks such as America On Line (AOL).

Entered into an agreement with a company (Viacrypt, now Network Associates) to provide a fully compatible, low-cost commercial version of PGP. Since its beginnings about 15 years ago, PGP has grown explosively and is now very widely used. A number of reasons are cited for such growth:

It is available for free worldwide in versions that run on many different platforms, including Windows, UNIX, Mac, etc.

In addition, the commercial version satisfies those who want vendor support. 117, Chapter 12, Pretty Good Privacy (PGP)

It is based on algorithms that have survived extensive public review and are considered secure. Specifically, the package includes RSA, DSS, and Diffie-Hellman for public-key encryption; CAST-128, IDEA, and 3DES for symmetric encryption; and SHA-1 for hash coding.

It has a wide range of applicability, from corporations that wish to select and enforce a standardized scheme for encrypting files and messages to individuals who wish to communicate securely with others worldwide over the Internet.

It was not developed by, nor is it controlled by, any government or standards organization. For those with an instinctive distrust of "the establishment", this makes PGP attractive. In the last few years, commercial versions have become available.

PGP is now on the Internet standards track (RFC 3156). Nevertheless, PGP still has the aura of an anti-establishment endeavor.

4.1.2.1 PGP Operations:

The following are the services offered by PGP:

1.Authentication

2.Confidentiality

3.Compression

4.Email Compatibility

1. Authentication:

Authentication basically means something that is used to validate something as true or real.

To login to some sites, sometimes we give our account name and password; that is an authentication verification procedure.

The Authentication service in PGP is provided as follows:

The Hash Function (H) calculates the Hash Value of the message. For hashing purposes, SHA-1 is used, which produces a 160-bit output hash value.

Then, using the sender's private key (KPa), it is encrypted, and it's called a digital Signature.

The Message is then appended to the signature.

At the receiver's end, the data is decompressed, and the message and signature are obtained.

The signature is then decrypted using the sender's public key (PUa), and the hash value is obtained.

Both the values, one from the signature and another from the recent output of the hash function, are compared, and if both are the same, it means that the email was actually sent from a known sender and is legit; otherwise, it means that it's not a legit one.

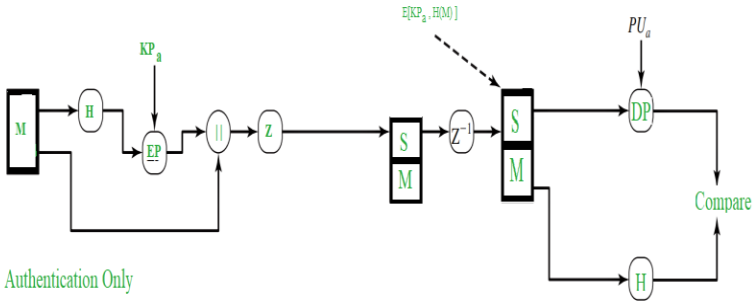

Figure 4.1: PGP Authentication. (Copy right is reserved to William Stallings.)

2. Confidentiality:

When it comes confidentiality, only the sender who is sending and the intended recipient should be able to see the message, which implies that everyone else must be kept

restricted from the content of that message. The confidentiality goes with PGP as shown in the figure……

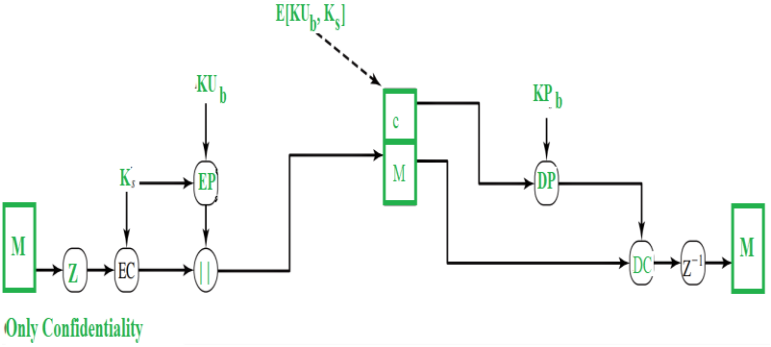

Figure 4.2: PGP Confidentiality. (Copy right is reserved to William Stallings.)

First, the message undergoes compression, and then a 128-bit session key (Ks) generated by PGP is employed for symmetric encryption to secure the message. Next, the session key (Ks) is encrypted using public key encryption (EP) with the receiver's public key (KUb). The encrypted session key and the encrypted message are combined and transmitted to the receiver.

Upon receipt, the receiver decrypts the encrypted session key using their private key (KPb) and subsequently decrypts the message using the obtained session key.

3.Compression:

Compression is a mechanism to reduce the amount of data to be transferred. Compression is basically converting a message of n bits to m bits (n > m) using a compression algorithm.

The compression service in PGP is provided using the ZIP Algorithm.

The Compression is included in the combined Authentication and Confidentiality of PGP as follows:

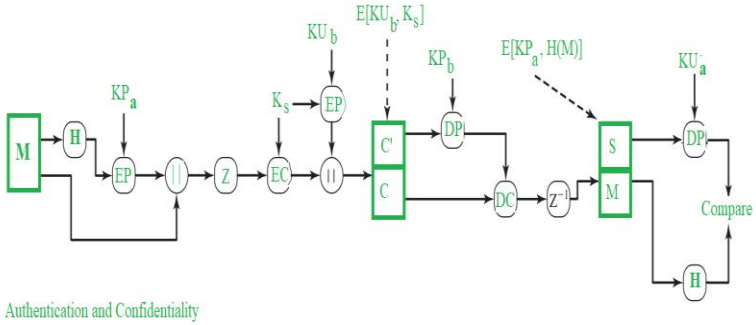

Fig:4.3 Compression

In the figure above, Z represents the compression function, and Z-1 represents the decompression function.

In PGP, messages are compressed only after the application of a signature.

The compressed data is decompressed at the receiver's end to obtain the original message and the signature.

4.Email Compatibility:

Many electronic mail systems only permit the use of blocks consisting of ASCII text.

When PGP is used, at least part of the block to be transmitted is encrypted.

This basically produces a sequence of arbitrary binary words, which some mail systems won't accept.

To accommodate this restriction, PGP uses an algorithm known as radix64, which maps 6 bits of binary data into an 8-bit ASCII character.

Unfortunately, this expands the message by 33%; however, with the compression algorithm, the overall compression will be about one third (in general).

4.1.2.2 Cryptographic Keys and Key Rings

PGP makes use of four types of keys:

- One-time session symmetric keys
- Public keys
- Private keys
- Passphrase-based symmetric keys

Three separate requirements can be identified with respect to these keys:

- A means of generating unpredictable session keys is needed.
- We would like to allow a user to have multiple public-key and private-key pairs.
- As a result, there is not a one-to-one correspondence between users and their public keys.

Thus, some means are needed for identifying particular keys.

4.1.2.3 PGP Operation: Summary:

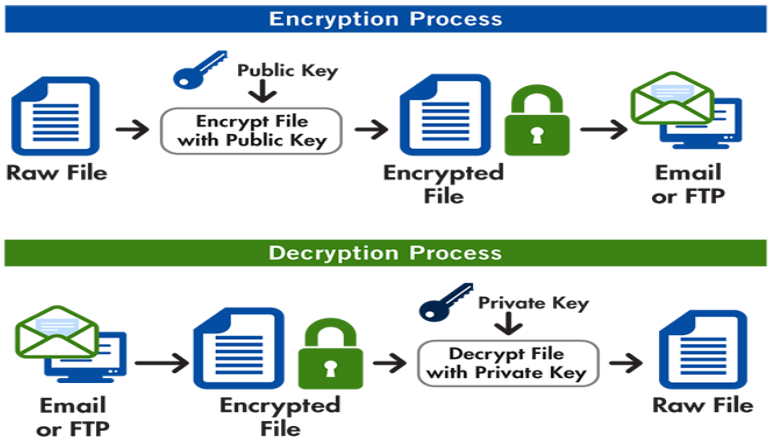

Figure4.4: PGP-Operations (Copy right is reserved to William Stallings.)

4.1.2.4 Key Identifiers:

As mentioned, it is possible to have more than one public/private key pair per user. Each key contains an ID.

- The key ID associated with each public key consists of its least significant 64 bits.
- That is, the key ID of public key KUa is (KUa mod 264).
- A key ID is also used for the PGP digital signature, as the sender may use one of a number of private keys to encrypt the message digest, and the recipient must know which one was used.
- The key ID concept can be understood by following the below figure.

PGP message format:

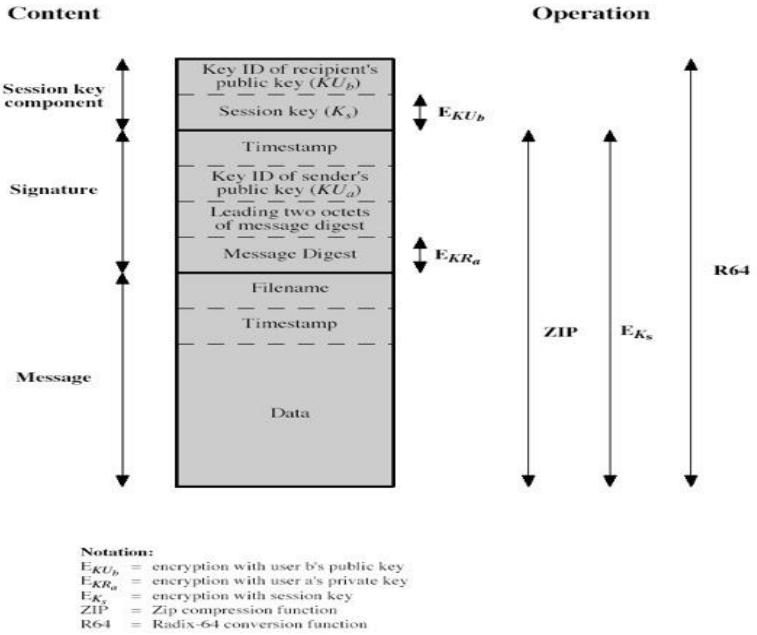

Figure4.5: PGP message format (Copy right is reserved to William Stallings.)

Key Rings:

Key IDs are critical to the operation of PGP. From figure.

These keys need to be stored and organized in a systematic way for efficient and effective use by all parties.

The scheme used in PGP is to provide a pair of data structures at each node, one to store the public and private key pairs owned by that node and one to store the public keys of other users known at this node.

These data structures are referred to, respectively, as the private-key ring and the public-key ring.

We can view the ring as a table where each row represents one of the public or private key pairs owned by this user. Each row contains the following:

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
T_1	$KU_i \bmod 2^{64}$	KU_i	$E_{H(P)}[KR_i]$	User i
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
T_1	$KU_i \bmod 2^{64}$	Ku_i	$trust\_flag_i$	User i	$trust\_flag_i$		
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•

* = field used to index table

Figure 4.6: Key Rings (Copy right is reserved to William Stallings.)

General structure of private and public-key rings

Timestamp: The date or time when this key pair was generated.

Key ID: The least significant 64 bits of the public key for this entry.

Public Key: The public-key portion of the pair

Private key: The private-key portion of the pair

4.1.2.5 PGP message generation:

PGP message generation (without compression or radix64 conversion) using all the terms we have met (reception is similar).

Figure 4.7 shows PGP Message generation (from User A to User B; no compression or radix64 conversion).

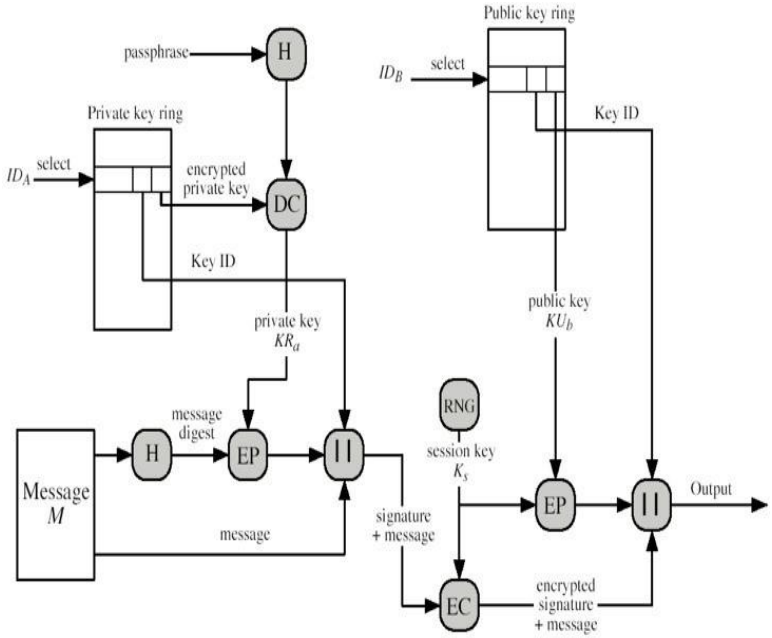

Figure4.7: PGP message generation (Copy right is reserved to William Stallings.).

4.1.3 S/MIME (Secure/Multipurpose Internet Mail Extensions):

What Is S/MIME?

S/MIME, or Secure/Multipurpose Internet Mail Extension, is a technology widely used by corporations that enhances email security by providing encryption, which protects the content of email messages from unwanted access.

It also adds digital signatures, which confirm that you are the authentic sender of the message, making it a powerful weapon against many email-based attacks.

S/MIME Uses:

S/MIME can be used to:

- Check that the email you sent has not been tampered with by a third party.
- Create digital signatures to use when signing emails.
- Encrypt all emails.
- Check the email client you're using.

S/MIME Functions:

S/MIME provides the following functions:

Enveloped data: This consists of encrypted content of any type and encrypted-content encryption keys for one or more recipients.

Signed data: A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer. The content plus signature is then encoded using base64 encoding. A signed data message can only be viewed by a recipient with S/MIME capability.

Clear-signed data: As with signed data, a digital signature of the content is formed. However, in this case, only the digital signature is encoded using base64. As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.

Signed and enveloped data: Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.

S/MIME Cryptographic Algorithms:

S/MIME (Secure/Multipurpose Internet Mail Extensions) is a protocol used for securing email communications through the use of digital certificates and cryptographic algorithms.

S/MIME uses various cryptographic algorithms to provide message confidentiality, integrity, authentication, and non-repudiation.

Some of the cryptographic algorithms used in S/MIME include:

- RSA (Rivest-Shamir-Adleman) encryption for key exchange and digital signatures
- DSA (Digital Signature Algorithm) for digital signatures
- AES (Advanced Encryption Standard) and 3DES (Triple Data Encryption)
- Standard) for symmetric encryption
- SHA (Secure Hash Algorithm) for message digest and hash functions

S/MIME Messages:

S/MIME secures a MIME entity with a signature, encryption, or both. A MIME-wrapped PKCS object has a range of content types:

- enveloped data.
- signed data.
- clear-signed data.
- registration request.
- certificate-only message.

S/MIME Certificate Processing:

An S/MIME certificate needs to be installed on the email clients of both the recipient and the sender to ensure email encryption at both ends.

When an email is sent, the sender encrypts the email using the recipient's public key, and the recipient decrypts the email using the recipient's private key.

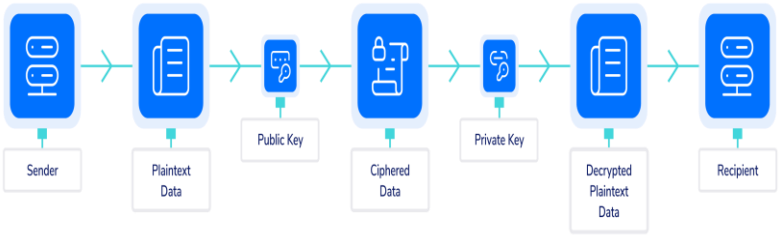

Figure 4.8: S/MIME Certificate Processing. (Copy right is reserved to William Stallings.)

Certificate Authorities:

have several well-known CA's, Verisign being one of the most widely used. Verisign issues several types of Digital IDs, increasing levels of checks, and hence trusting class Class check usage.

1 name/email check web browsing/email.

2 + enrol/addr check email, subs, s/w validate.

3 + ID documents e-banking/service access.

In a nutshell, S/MIME is a commonly used protocol for sending encrypted and digitally signed email.

4.2 Working Principle of Domain Key-Identified Mail.

What is DKIM?

Domain Keys Identified Mail (DKIM) is an email security standard that ensures messages are not modified while travelling between the recipient and sending servers. DKIM permits organizations to take responsibility for transmitting a message in a way a recipient can verify.

The organization can be the originating website, an intermediary, etc. Their reputation is the basis for evaluating whether or not to trust the message's delivery.

4.2.1 What is a DKIM Signature?

DKIM gives emails a signature header that is added to the email and secured with encryption.

Each DKIM signature contains all the information needed for an email server to verify that the signature is real, and a pair of DKIM keys encrypt it.

The originating email server has the 'private DKIM key,' which can be verified by the receiving mail server or ISP with the other half of the keypair, called the 'public DKIM key.'

These signatures travel with the emails and are verified along the way by the email servers that move the emails toward their final destination.

4.2.2 Why should you authenticate emails with DKIM?

DKIM is among the top three must-have authentication protocols, and that's for good reason as it impacts the deliverability and reputation of your domain.

Some of the reasons behind getting your domain DKIM authenticated are as follows:

Maintains the legitimacy of the sender.

When you have a DKIM-signed email, it signals to ISPs that the email is not tampered with.

This helps improve your reputation as a sender, as it seems legitimate to the receiver's server.

The better your sender's reputation, the more you'll land in the recipient's inbox, leading to higher deliverability.

Protect your domain and secure your email.

DKIM prevents email spoofing sent from your domain. When DKIM authenticates your emails, every outgoing message will have the DKIM signature.

This signature will ensure that the email content is not tampered with. Hence, keep your email away from spoofed domains.

Get in the good books of ISPs.

ISPs are the major contributing factor in deciding where to land your email.

Having your emails signed by DKIM is one of the best ways to catch their attention and show them that you are a verified sender, and the content is not tampered with.

The more you show ISPs that your content and sending domain are genuine, the more you'll see your emails land in your inbox.

4.2.3 How does DKIM work?

DKIM is added as a signature to your email's header once it's verified by both the sender's and receiver's servers.

The signature is verified using a private and public cryptographic key.

The private key is safe and hosted on your server or your ESP. As it's private, only you can have access to it.

To validate the DKIM signature, ISPs look at the public key hosted on your organization’s DNS record.

This record is public and can be accessed by anyone to verify the legitimacy of your sender's domain.

The process looks like this:

A domain owner publishes a cryptographic public key as a specially formatted TXT record in the domain's overall DNS records.

On the sender's server end.

When an email is sent, the domain generates a private key.

This key contains all the information the mail server needs to verify your messages.

On the receiver's server end.

When the recipient server receives a DKIM-signed email, it uses the public key published to DNS to check the source message and the message body.

It is done to check if any changes were made during the transit.

Once the recipient server verifies the signature with the public key, the message is deemed authentic. After that, it is passed on to the ESP.

4.2.4 Verification of DKIM-Signed Messages

For any email, the signature from DKIM is like a tamper-proof seal.

The signature shows that it has come from the original domain and hasn't been tampered with.

Every email sent is attached with a signature that is specially configured on the email servers to use DKIM.

4.2.5 What is the DKIM record?

When an email is sent, there may be an intervention in the pipeline by a hacker.

This may result in the email content being forged. So, to address this issue, anti-spam bodies have conceptualized the DKIM record.

This happens with the help of a simple encoding and decoding method, as follows:

Sender level

The email combines with a private key available in the sender server software to form an email signature.

This signature is sent to the recipient along with the actual message.

Recipient level

There's a Domain Name System (DNS) record published in the world, available as the public key.

The DNS record looks as follows:

- How do you set up your own DKIM key?
- Is Mailmodo compatible with DKIM?

Yes, Mailmodo is compatible with DKIM. Our email specialists team helps you understand the importance of security certifications and build your IP and domain reputation for high email deliverability.

We help our users set up DKIM without any hassle to help them get whitelisted for AMP email from email clients.

4.3 Understand and apply the IP Security Policy.

4.3.1 IP Security

IP Sec (Internet Protocol Security) is an Internet Engineering Task Force (IETF) standard suite of protocols between two communication points across the IP network that provide data authentication, integrity, and confidentiality.

It also defines the encrypted, decrypted, and authenticated packets.

The protocols needed for secure key exchange and key management are defined in it.

Uses of IP Security – IP Sec can be used to do the following things:

- To encrypt the application layer
- To provide security for routers sending routing data across the public internet.
- To provide authentication without encryption, like to authenticate that the data originates from a known sender.
- To protect network data by setting up circuits using IPsec tunnelling in which all data being sent between the two endpoints is encrypted, as with a Virtual Private network (VPN),

4.3.2 Components of IP Security

It has the following components:

- Encapsulating Security Payload (ESP)
- Authentication Header (AH)
- Internet Key Exchange (IKE)

1. Encapsulating Security Payload (ESP):

It provides data integrity, encryption, authentication, and anti-replay. It also provides authentication for the payload.

2. Authentication Header (AH):

It also provides data integrity, authentication, and anti-replay, but it does not provide encryption. The anti-replay protection protects against the unauthorised transmission of packets. It does not protect data confidentiality.

3. Internet Key Exchange (IKE):

It is a network security protocol designed to dynamically exchange encryption keys and find a way over the Security Association (SA) between two devices.

4.3.3 IP Security Architecture

The IPsec (IP Security) architecture uses two protocols to secure the traffic or data flow.

These protocols are ESP (Encapsulation Security Payload) and AH (Authentication Header).

IPsec Architecture includes protocols, algorithms, DOIs, and Key Management.

All these components are very important in order to provide the three main services:

1. Confidentiality
2. Authenticity
3. Integrity

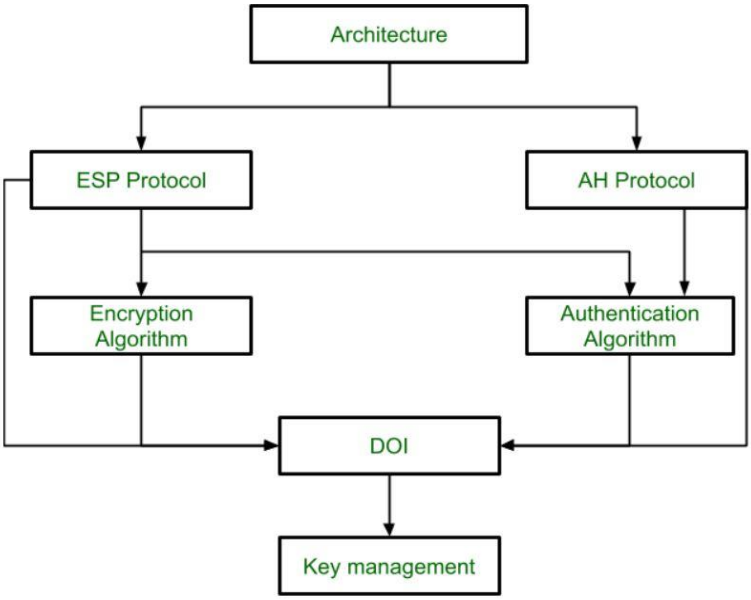

Figure 4.8: IP Sec (Internet Protocol Security) (Copy right is reserved to William Stallings.)

4.3.4 Features of IP Security.

Authentication: IP Sec provides authentication of IP packets using digital signatures or shared secrets. This helps ensure that the packets are not tampered with or forged.

Confidentiality: IP Sec provides confidentiality by encrypting IP packets, preventing eavesdropping on network traffic.

Integrity: IP Sec provides integrity by ensuring that IP packets have not been modified or corrupted during transmission.

Key management: IP Sec provides key management services, including key exchange and key revocation, to ensure that cryptographic keys are securely managed.

Tunnelling: IP Sec supports tunnelling, allowing IP packets to be encapsulated within another protocol, such as GRE (Generic Routing Encapsulation) or L2TP (Layer 2 Tunnelling Protocol).

Flexibility: IPSec can be configured to provide security for a wide range of network topologies, including point-to-point, site-to-site, and remote access connections.

Interoperability: IP Sec is an open standard protocol, which means that it is supported by a wide range of vendors and can be used in heterogeneous environments.

Securing Internet Protocol (IP) Storage: Storage networking technology has enjoyed strong growth in recent years, but security concerns and threats facing networked data have grown equally fast. Today, there are many potential threats that are targeted at storage networks, including data modification, destruction, and theft, DoS attacks, malware, hardware theft, and unauthorised access, among others. In order for a Storage Area Network (SAN) to be secure, each of these threats must be individually addressed. In this paper, we present a comparative study by implementing different security methods in an IP Storage network.

4.4 Encapsulating Security Payload (ESP).

The Encapsulating Security Payload (ESP) protocol encrypts and authenticates data packets sent between computers via a virtual private network (VPN). VPNs can work securely because of the emphasis and layers under which ESP functions.

The Encapsulating Security Payload (ESP) protocol provides:

1. Data confidentiality
2. Data origin authentication
3. Data integrity.
4. Replay protection.

4.4.1 ESP Architecture

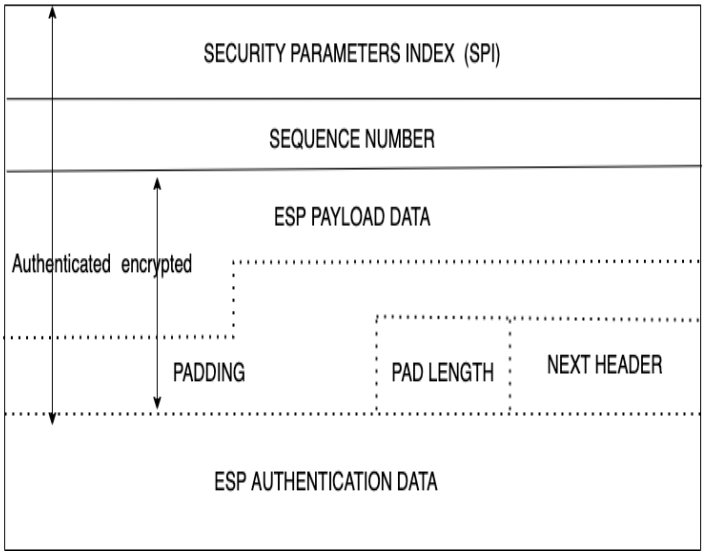

Figure4.9: ESP Architecture. (Copy right is reserved to William Stallings.)

Security parameter index (SPI): The **SPI** is a 32-bit value that, when combined with the packet's destination IP address and security protocol, uniquely identifies the Security Association (SA).

Sequence number: The sequence number is a 32-bit counter that increases monotonically to protect against replay attacks. The sequence number is reset to 0 when a SA is established. On the sender's and receiver's ends, it is first set to 0. As packets move from sender to receiver, the counter is incremented. Finally, the counter is checked on the receiver's side.

ESP payload data: ESP payload data is a transport-level segment or IP packet that is protected by encryption. This is where our actual message resides, and it is encrypted for

confidentiality. This is a variable-length field that normally holds the data payload.

Padding: It is used to fill the payload data to a specific block size multiple required by a specific encryption scheme or to randomize the length of the payload in order to protect it against traffic.

Pad length: It is an 8-bit field whose value shows the padding field's length in bytes.

Next header: The next header identifies the type of data contained in the payload data field by identifying the first header in that payload (e.g., an extension header in IPv6 or an upper-layer protocol such as TCP).

ESP authentication data: It is a variable-length field containing the integrity check value (ICV). **ICV** verifies the sender's identity and the integrity of the message. ICV is an optional field.

4.4.2 Encryption and Authentication Algorithms and Protocols:

ESP can encrypt payload data, padding, pad length, and the next header.

- If needed, have IV at the start of the payload data.
- ESP can have an optional ICV for integrity.
- is computed after encryption is performed.
- ESP uses padding.
- to expand plaintext to the required length.
- to align pad length and next header fields

- to provide partial traffic flow confidentiality.

4.5 The concept of Firewalls and Gateways.

4.5.1 Fundamentals of Firewalls.

Firewalls are an essential component of network security, serving as the first line of defense against cyber threats. In this subchapter, we will delve into the fundamentals of firewalls, their purpose, and their role in protecting critical infrastructure networks.

A Firewall is a network security device that monitors and controls incoming and outgoing network traffic based on predetermined security rules. Its main objective is to create a barrier between trusted internal networks and untrusted external networks, such as the internet, to prevent unauthorized access and potential cyber-attacks.

Firewalls operate at deferent levels of the network stack, including the network, transport, and application layers, to filter and inspect network traffic. Network firewalls, also known as packet filters, examine packets based on source and destination IP addresses, port numbers, and protocols. Transport layer firewalls, such as circuit-level gateways, operate at the transport layer (e.g., TCP, UDP) to ensure the integrity of connections. Application layer firewalls, also known as proxy firewalls, analyze the content of network traffic to detect and block malicious activities.

One of the key features of a firewall is the ability to create and enforce access control policies. These policies define what traffic is allowed or denied based on specific criteria, such as IP addresses, port numbers, and protocols. By configuring firewall rules, administrators can permit or restrict traffic flow, ensuring only authorized users and applications can access the network.

Firewalls also provide additional security functionalities, such as Network Address Translation (NAT), which masks internal IP addresses from external networks, and Virtual Private Network (VPN) support, which allows secure remote access to the network.

4.5.2 Firewall design goals.

All traffic between the internal and external networks must pass through the firewall, which is achieved by restricting access to the local network, except through the firewall. Various configurations are possible, as discussed later.

Only authorized traffic, according to the local security policy, is permitted to traverse the firewall. Different types of firewalls are used to implement diverse security policies.

The firewall itself is made impervious to penetration, requiring the use of a trusted system with a secure operating system.

Additionally, firewalls utilize four general techniques to control access and enforce the site's security policy. Originally, firewalls focused mainly on service control, but they have evolved to encompass all four techniques:

Service control: This technique determines the types of Internet services that can be accessed, whether inbound or outbound. The firewall may use filtering based on IP address and TCP port number, proxy software to interpret service requests, or even host the server software itself (e.g., Web or mail services).

Direction control: Direction control dictates the allowed direction in which specific service requests can be initiated and permitted to flow through the firewall.

User control: User control manages access to services based on the user attempting to access them. This feature is usually applied to users within the firewall perimeter (local users) and may also extend to incoming traffic from external users, requiring secure authentication technology like IP Sec.

Behavior control: This technique governs how specific services are utilized. For instance, the firewall may filter email to eliminate spam or provide external access to only a portion of the information on a local Web server.

These combined techniques contribute to the firewall's ability to regulate network traffic, enhance security, and protect the organization's resources and sensitive information effectively.

4.5.3 Types of Firewalls

The following are popular types of Firewalls.

1. Packet Filtering Firewalls:

Packet filtering firewalls are the most basic type of firewall and operate at the network layer of the OSI model. They examine individual packets of data and make filtering decisions based on pre-defined rules. These firewalls analyze source and destination IP addresses, ports, and protocols to determine whether to allow or deny traffic. Configuring packet filtering firewalls involves defining access control lists (ACLs) and configuring rules based on specific criteria.

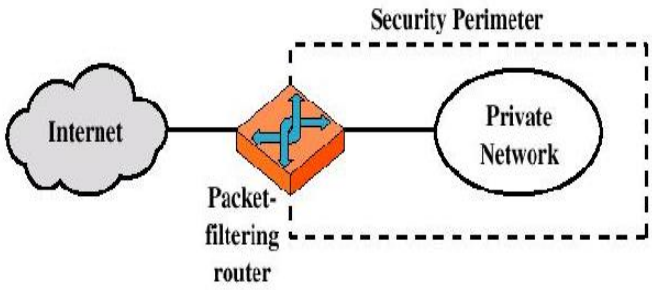

Figure 4.10: Packet filtering Firewall. (Copy right is reserved to William Stallings.)

Advantages:

- Simplicity
- Transparency to users
- High speed

Disadvantages:

- Difficulty of setting up packet filter rules
- Lack of Authentication
- Possible attacks and appropriate countermeasures
- IP address spoofing

- Source routing attacks
- Tiny fragment attacks

2. **Stateful Inspection Firewalls:**

Stateful inspection firewalls operate at the network and transport layers of the OSI model. They not only analyze individual packets but also keep track of the state of connections. Stateful inspection firewalls maintain a state table, which tracks the progress of network connections, enabling them to make more informed filtering decisions. Configuring stateful inspection firewalls involves defining rules based on packet attributes and connection states.

3. **Application-Level Gateways (Proxy Firewalls):**

Application-level gateways, also known as proxy firewalls, operate at the application layer of the OSI model. They act as intermediaries between clients and servers, intercepting and filtering all incoming and outgoing traffic. Proxy firewalls inspect the entire application layer payload, allowing for more granular control and enhanced security. Configuring proxy firewalls involves setting up specific proxies for each application or network service.

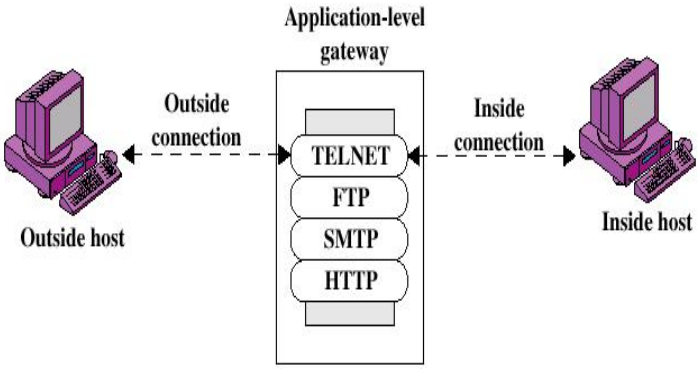

Figure4.11: Application-level gateways. (Copy right is reserved to William Stallings.)

Advantages:

- Higher security than packet filters Only need to scrutinize a few allowable applications.
- Easy to log and audit all incoming traffic.

Disadvantages:

- Additional processing overhead on each connection (gateway as splice point)

4. Next-Generation Firewalls (NGFWs):

Next-generation firewalls integrate traditional firewall functionality with additional security features, such as intrusion prevention systems (IPS), anti-malware, and application awareness. NGFWs use deep packet inspection (DPI) to analyze traffic at the application layer, providing advanced threat detection and prevention capabilities. Configuring NGFWs involves defining rules based on applications, users, and content.

5. Bastion Host

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 or circuit-level gateway.

4.6 Intrusion Detection System.

4.6.1 Introduction:

Hardware and/or software Attempts to detect Intrusions Heuristics /Statistics Signatures Gathers and reports incidents Sent to the console Trigger a response Composition of an IDS Components are added to an existing network. Sensor copies a record of all network activity and sends it to the collector. Determines if an attack is taking place. Manager Laptop or Desktop with IDS Software Check for alerts. Change settings Databases house network baseline data or attack signatures.

Anomaly-Based vs. Signature-Based IDS

Anomaly Based Monitors network traffic and keeps track of patterns of traffic and information to obtain baseline If a deviation in network behaviour is detected, IDS will assume an attack. Higher risk of false positives signature-based attack A signature database is maintained. Compare traffic to the database. If a match is found, an alert is sent. Requires constant updates.

Network-Based vs. Host-Based IDS

Network-Based Monitors monitor all traffic on the network. Useful for monitoring non-critical systems. Host-Based IDs customized to a specific server Being closer to the host allows for a greater chance of detection. Prevents threats such as Trojans and backdoors from being installed within the network.

Passive vs. Reactive

Passive: When an attack is detected, an alarm or alter will be triggered; no further action is performed by the IDS.

Reactive: The Collector will send an alert. Send instructions to the firewall and router to block activity from occurring on the network. Responses should be managed and assessed, regardless of the system being used.

4.6.2 Challenges of IDS:

False Negatives

When an IDS fails to detect an attack, False negatives occur when the pattern of traffic is not identified in the signature database, such as new attack patterns. False negatives are deceptive because you usually have no way of knowing if and when they occurred.

You are most likely to identify false negatives when an attack is successful but isn’t detected by the IDS.

False Positives

Described as a false alarm. When an IDS mistakenly reports certain "normal" network activity as malicious. Administrators have to fine-tune the signatures or heuristics in order to prevent this type of problem.

Why IDS important?

The ability to know when an intruder or attacker is engaged in reconnaissance or other malicious activity can mean the difference between being compromised and not being compromised. An IDS can alert the administrator of a successful compromise, giving them the opportunity to implement mitigating actions before further damage is caused. As Corporations and other Institutions are legally compelled to disclose data breaches and compromises to their affected customers, this can have profound effects on a compromised company in the way of bad press, loss of customer trust, and effects on their stock.

4.6.3 How does it fit into our security plan?

As a network security expert, we should know that we cannot just rely on one or a few tools to secure our network. we need to have a defense-in-depth mindset and layer our network defenses. By using inside and outside firewalls, DMZs, routers, and Switches, an IDS is a great addition to our security plan.

we can use them to identify vulnerabilities and weaknesses in our perimeter protection devices, such as firewalls, switches, and routers. The firewall rules and router access control lists can be verified regularly for compliance.

we can use IDS to enforce security policies, such as unauthorized Internet access, downloads of executable files, use of file sharing programs like Kazza, or Instant Messenger use. IDPs are also an invaluable source of evidence. Logs from an IDS can become an important part of computer forensics and incident handling efforts.

4.6.4 Pros & Cons of IDS.

Pros

- Can detect external hackers as well as internal network-based attacks.
- Scales easily to provide protection for the entire network.
- Offers centralized management for the correlation of distributed attacks Provides defense in depth.
- Gives administrators the ability to quantify attacks. Provides an additional layer of protection.

Cons

- Skilled staff dedicated to interpreting the data Requires a complex incident response process.
- Cannot monitor traffic at higher network traffic rates. Generates an enormous amount of data to be analyzed.
- Cannot deal with encrypted network traffic.

4.7 Terminal Questions

1. Explain PGP and its Operations?
2. Explain S/MIME function with an example.
3. Explain key rings?
4. Explain S/MIME Cryptographic Algorithms?
5. Explain S/MIME Messages?
6. Explain how to set up a DMV key.
7. Explain the DKIM record.
8. Explain how to authenticate emails with DKIM.
9. Explain the DKIM Signature?
10. What are some common security measures for network management?

11. How can I enhance the security of my devices?
12. What is a VPN and how does it work?
13. Can you recommend any reliable VPN providers?
14. How can I ensure the security of my VPN connection?
15. Explain IP security and its uses.
16. Explain encryption and authentication algorithms and padding.
17. Explain the Encapsulating Security Payload (ESP)?
18. Explain IP Security architecture.
19. Explain the types of Firewalls.
20. Explain the characteristics of Fire walls.
21. Explain the Firewall design principles.
22. List out pros and cons of IDS.
23. Explain about the different types IDPs.

Phone 7036844638

Email: dsekharbabu@kluniversity.in

https:kluniversity.in.

References

1. "Computer Networking: A Top-Down Approach" by James F. Kurose and Keith W. Ross.
2. "Data Communications and Networking" by Behrouz A. Forouzan.
3. "Network Warrior: Everything You Need to Know That Wasn't on the CCNA Exam" by Gary A. Donahue.
4. "TCP/IP Illustrated, Volume 1: The Protocols" by W. Richard Stevens.
5. "Network Security Essentials: Applications and Standards" by William Stallings.
6. "Computer Security: Principles and Practice" by William Stallings and Lawrie Brown.
7. "The Art of Computer Virus Research and Defense" by Peter Szor.
8. "Hacking Exposed: Network Security Secrets and Solutions" by Stuart McClure, Joel Scambray, and George Kurtz.
9. "Securing the Internet of Things: A Practical Approach to Device Security" by Shancang Li, Li Da Xu, and Liming Chen.
10. "Virtual Private Networks (VPNs): A Beginner's Guide" by John Mairs.
11. "Network Management: Principles and Practices" by Mani Subramanian.
12. "Wi-Foo: The Secrets of Wireless Hacking" by Andrew Vladimirov, Konstantin Gavrilenko, and Andrei Mikhailovsky.

13. "Hacking Exposed Wireless: Wireless Security Secrets & Solutions" by Johnny Cache, Joshua Wright, and Vincent Liu.
14. "CWNA Certified Wireless Network Administrator Study Guide" by David D. Coleman and David A. Westcott.
15. CCNA Security 210-260 Official Cert Guide" by Omar Santos and John Stuppi.
16. "Network Security Technologies and Solutions" by Yusuf Bhaiji.
17. "Computer Networks: A Systems Approach" by Larry L. Peterson and Bruce S. Davie.
18. "High-Performance Communication Networks" by Jean Walrand and Pravin Varaiya.
19. "Quality of Service in Packet Networks: Basic Mechanisms and Directions" by Malathi Veeraraghavan.
20. "Queueing Systems: Theory and Applications" by Leonard Kleinrock and Richard Gail.
21. Cryptography and Network Security Principles and Practice, by William Stallings, Pearson, 7th edition, 2017.
22. Cryptography and Network Security by Behrouz A. Forouzan, Debdeep Mukhopadhyay, Tata McGraw Hill Education Private Limited, Fourth edition 2015.
23. http://mercury.webster.edu/aleshunas/COSC%205130/Chapter-22.pdf
24. http://www.uky.edu/~dsianita/390/firewall1.pdf
25. https://www.linkedin.com/learning/it-security-foundations-network-security

26. https://www.linkedin.com/learning/it-security-foundations-core-concepts

27. https://www.coursera.org/specializations/computer-network-security

(DEEMED TO BE UNIVERSITY)

KONERU LAKSHMAIAH EDUCATION FOUNDATION,
GREEN FIELDS, VADDESWARAM,
GUNTUR-522502

www.kluniversity.in