R.M.K GROUP OF ENGINEERING INSTITUTIONS

R.M.K
GROUP OF
INSTITUTIONS

R.M.K GROUP OF INSTITUTIONS

R.M.K
GROUP OF
INSTITUTIONS

R.M.K
GROUP OF
INSTITUTIONS

Please read this disclaimer before proceeding:

This document is confidential and intended solely for the educational purpose of RMK Group of Educational Institutions. If you have received this document through email in error, please notify the system manager. This document contains proprietary information and is intended only to the respective group / learning community as intended. If you are not the addressee you should not disseminate, distribute or copy through e-mail. Please notify the sender immediately by e-mail if you have received this document by mistake and delete this document from your system. If you are not the intended recipient you are notified that disclosing, copying, distributing or taking any action in reliance on the contents of this information is strictly prohibited.

22CY701
INTRUSION DETECTION AND INTERNET SECURITY
(Lab Integrated)

UNIT II
INTRUSION DETECTION AND PREVENTION TECHNIQUES

Department	**: CSE(CS)**
Batch/Year	**: 2022 - 2026 /IV**
Created by	**:Dr. Dharini N**
Date	**: 10.08.2025**

R.M.K
GROUP OF
INSTITUTIONS

Table of Contents

Course Objectives

22CY701 INTRUSION DETECTION AND INTERNET SECURITY
(Lab Integrated)

COURSE OBJECTIVES

- To Understand when, where, how, and why to apply Intrusion Detection tools and techniques in order to improve the security posture of an enterprise.
- To Apply knowledge of the fundamentals and history of Intrusion Detection in order to avoid common pitfalls in the creation and evaluation of new Intrusion Detection Systems
- To Analyze intrusion detection alerts and logs to distinguish attack types from false alarms
- To Understand the fundamentals of network security, including the MAC layer, Internet Protocol, and common attacks targeting these layers.
- To Gain an understanding of key internet security mechanisms, including firewalls, virtual private networks (VPNs), and TLS/SSL VPNs.

Prerequisite

22CY701 INTRUSION DETECTION AND INTERNET SECURITY

PREREQUISITE

1. 22CY401-CYBER SECURITY ESSENTIALS
2. 22CS501 – COMPUTER NETWORKS
3. 22CS901 – ETHICAL HACKING

Syllabus

22CY701 – INTRUSION DETECTION AND INTERNET SECURITY (Lab Integrated)

SYLLABUS **3 0 2 4**

UNIT I INTRODUCTION TO INTRUSION DETECTION

History of Intrusion detection, Audit, Concept and definition, Internal and external threats to data, attacks, Need and types of IDS, Information sources Host based information sources, Network based information sources.

List of Exercise/Experiments

1. Install Snort and configure it to monitor network traffic.

2. Deploy Snort as a Network Intrusion Detection System (NIDS).

UNIT II INTRUSION DETECTION AND PREVENTION TECHNIQUES

Intrusion Prevention Systems, Network IDs protocol based IDs , Hybrid IDs, Analysis schemes, thinking about intrusion. A model for intrusion analysis, techniques Responses requirement of responses, types of responses mapping responses to policy Vulnerability analysis, credential analysis non credential analysis

List of Exercise/Experiments

1. Write and implement custom Snort rules to detect specific traffic patterns.

2. Integrate Snort with MySQL to log alerts to a database.

UNIT III SNORT

Introduction to Snort, Snort Installation Scenarios, Installing Snort, Running Snort on Multiple Network Interfaces, Snort Command Line Options. Step-By-Step Procedure to Compile and Install Snort Location of Snort Files, Snort Modes Snort Alert Modes-Working with Snort Rules, Rule Headers, Rule Options, The Snort Configuration File etc. Plugins, Preprocessors and Output Modules, Using Snort with MySQL

List of Exercise/Experiments

1.Enhance Snort's functionality using preprocessors and plugins.

2.Set up advanced alerting and logging mechanisms.

UNIT IV ESSENTIALS OF INTERNET SECURITY

Network Security basics-The MAC Layer and Attacks- The Internet Protocol and Attacks- Packet Sniffing and Spoofing-Attacks on TCP Protocol- DNS Attacks Overview -Local DNS Cache Poisoning Attack- Remote DNS Cache Poisoning attack- Replay forgery attacks-DNS Rebinding attack- DoS on DNS Servers- DNSSEC-Securing DNS

List of Exercise/Experiments

1.Conducting TCP SYN Flood Attack

2.Sniffing Packets on Network Interfaces

3.Spoofing Source IP Address in Packets

4.Investigating DNSSEC Implementation and Validation

UNIT V INTERNET SECURITY MECHANISMS

Firewall-Virtual Private Network-Overview of How TLC/SSL VPN Works-Creating and using the TUN Interface- Implementing the IP Tunnel- Testing VPN- Tunneling and Firewall Evasion- -BGP and Attacks- The Heartbleed bug and attack- Reverse Shell

List of Exercise/Experiments

1. Configuring Linux Firewall using IP tables
2. Setting Up VPN Tunnels
3. Exploring BGP Session Hijacking
4. Simulating Heartbleed Attack Scenario

Course Outcomes

COURSE OUTCOMES

- CO1: Understand fundamental concepts and demonstrate skills in capturing and analyzing network packets.
- CO2: Utilize various protocol analyzers and Network Intrusion Detection Systems (NIDS) to detect network attacks and troubleshoot network problems.
- CO3: Develop the ability to proficiently use the Snort tool for detecting and mitigating network attacks
- CO4: Demonstrate knowledge of network security basics, including MAC layer vulnerabilities and attacks, as well as common attacks targeting the Internet Protocol.
- CO5: Demonstrate understanding of firewall, VPN, and TLS/SSL VPN principles and functionalities in network security.
- CO6: Apply the concepts of Intrusion Detection and internet security protocols to develop cyber security mechanisms.

CO – PO/ PSO Mapping

CO-PO MAPPING

COs	PO's/PSO's														
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	3	3	1	2	2	–	–	–	–	1	–	2	2	3	1
CO2	3	3	1	2	2	–	–	–	–	1	–	2	2	3	2
CO3	2	2	2	1	3	–	–	1	1	2	1	2	1	3	3
CO4	3	2	1	1	1	–	–	2	–	1	–	3	1	3	1
CO5	3	2	1	1	2	1	–	2	1	2	1	2	1	3	2
CO6	2	2	3	2	3	1	–	2	2	2	2	3	3	3	3

1 – Low, 2 – Medium, 3 – Strong

Lecture Plan

LECTURE PLAN

S No	Topics	No of periods	Proposed date	Actual Lecture Date	Pertaining CO	Taxonomy level	Mode of delivery
1	Network Security basics	1			CO4	K1	ICT Tools
2	The MAC Layer and Attacks	1			CO4	K2	ICT Tools
3	The Internet Protocol and Attacks	1			CO4	K3	ICT Tools
4	Packet Sniffing and Spoofing	1			CO4	K2	ICT Tools
5	Attacks on TCP Protocol	1			CO4	K2	ICT Tools
6	DNS Attacks Overview	1			CO4	K2	ICT Tools
7	Local DNS Cache Poisoning Attack	1			CO4	K2	ICT Tools
8	Remote DNS Cache Poisoning attack	1			CO4	K2	ICT Tools
9	Replay forgery attacks	1			CO4	K3	ICT Tools
10	DNS Rebinding attack	1			CO4	K3	Lecture and Practical
11	- DoS on DNS Servers	1			CO4	K3	Lecture and Practical
12	Write and DNSSEC-Securing DNS- Investigating DNSSEC Implementation and Validation	1			CO4	K3	Lecture and Practical
13	Conducting TCP SYN Flood Attack	1			CO4	K3	Lecture and Practical

14	Sniffing Packets on Network Interfaces	1			CO4	K3	Lecture and Practical
15	Spoofing Source IP Address in Packets	1			CO4	K3	Lecture and Practical

Activity Based Learning

Case Study Debate Topic: The Kaminsky DNS Cache Poisoning Attack (2008)

Lecture Notes

UNIT IV

1. NETWORK SECURITY BASICS

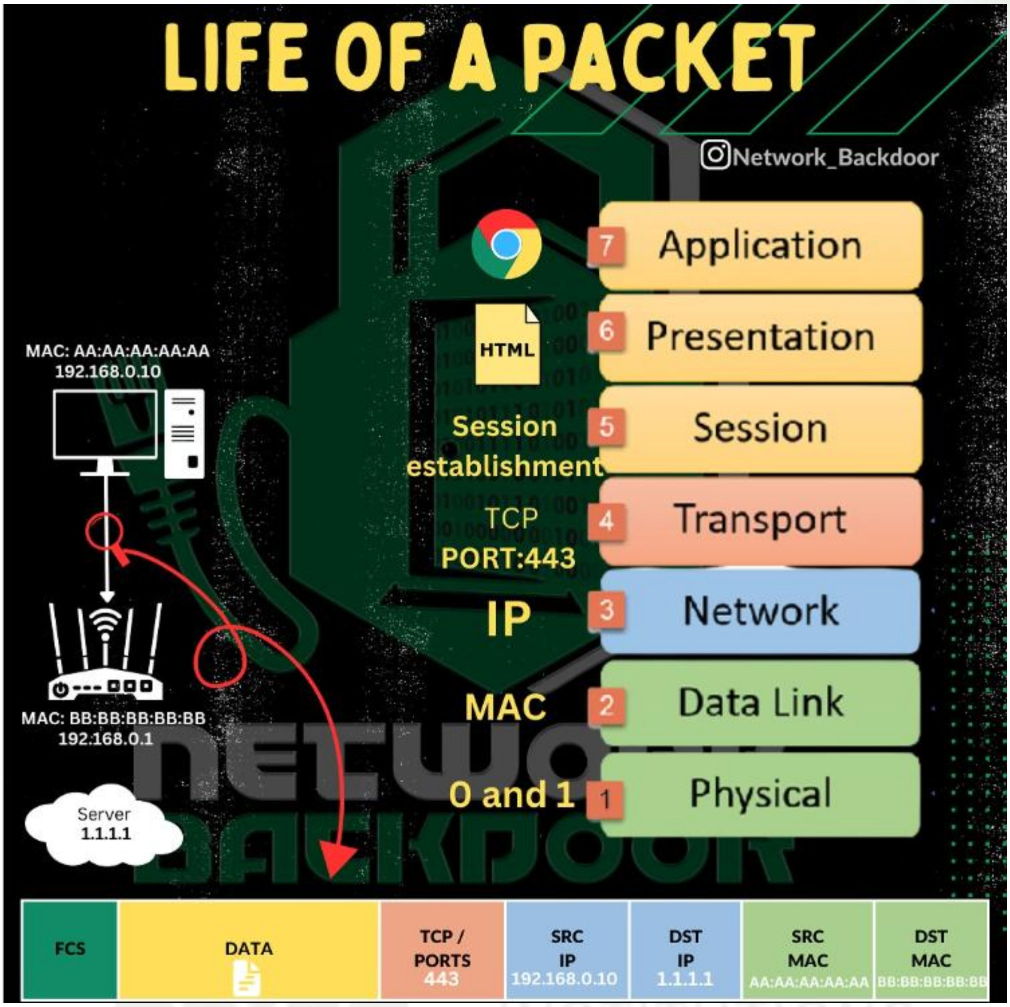
The Life-Cycle of Packet and Protocol Layers

The life-cycle of a packet begins at the application layer and travels down through the protocol stack before transmission. At a high level, a packet's journey starts when an application generates data (for example, a web browser sending an HTTP request). This data is passed to the transport layer, where a TCP or UDP header is added to provide port information and reliability features. The packet then moves to the network layer, where the IP header is added, containing source and destination IP addresses for routing. Finally, the link layer (e.g., Ethernet or Wi-Fi) encapsulates the packet with a MAC header before it is transmitted onto the physical medium.

When sending packets, the process involves system calls from applications to the operating system's kernel. The kernel network stack constructs the packet step by step, adding the necessary headers at each protocol layer. The packet is then handed off to the Network Interface Card (NIC), which physically transmits the data across the network. Inside the kernel, encapsulation ensures correct addressing, error detection, and compliance with networking protocols.

On the receiving end, the NIC captures incoming packets and delivers them to the kernel. The kernel removes each layer’s header in the reverse order: the MAC header, then the IP header, followed by the transport layer header. Once the packet reaches the transport layer, it is directed to the correct application socket based on the port number. Finally, the payload (application data) is delivered to the receiving program.

Sometimes, if a packet is not destined for the local machine, the system can act as a router. In this case, the kernel uses the routing table to determine the next hop for the packet. Routing decisions are based on IP addresses and subnet masks, allowing packets to travel across multiple networks until they reach their destination. To interact with packets, various packet-sending tools exist. For instance, ping is used to send ICMP echo requests, traceroute traces the path packets take across networks, netcat allows flexible TCP/UDP communication, and Scapy provides Python-based packet crafting and analysis.



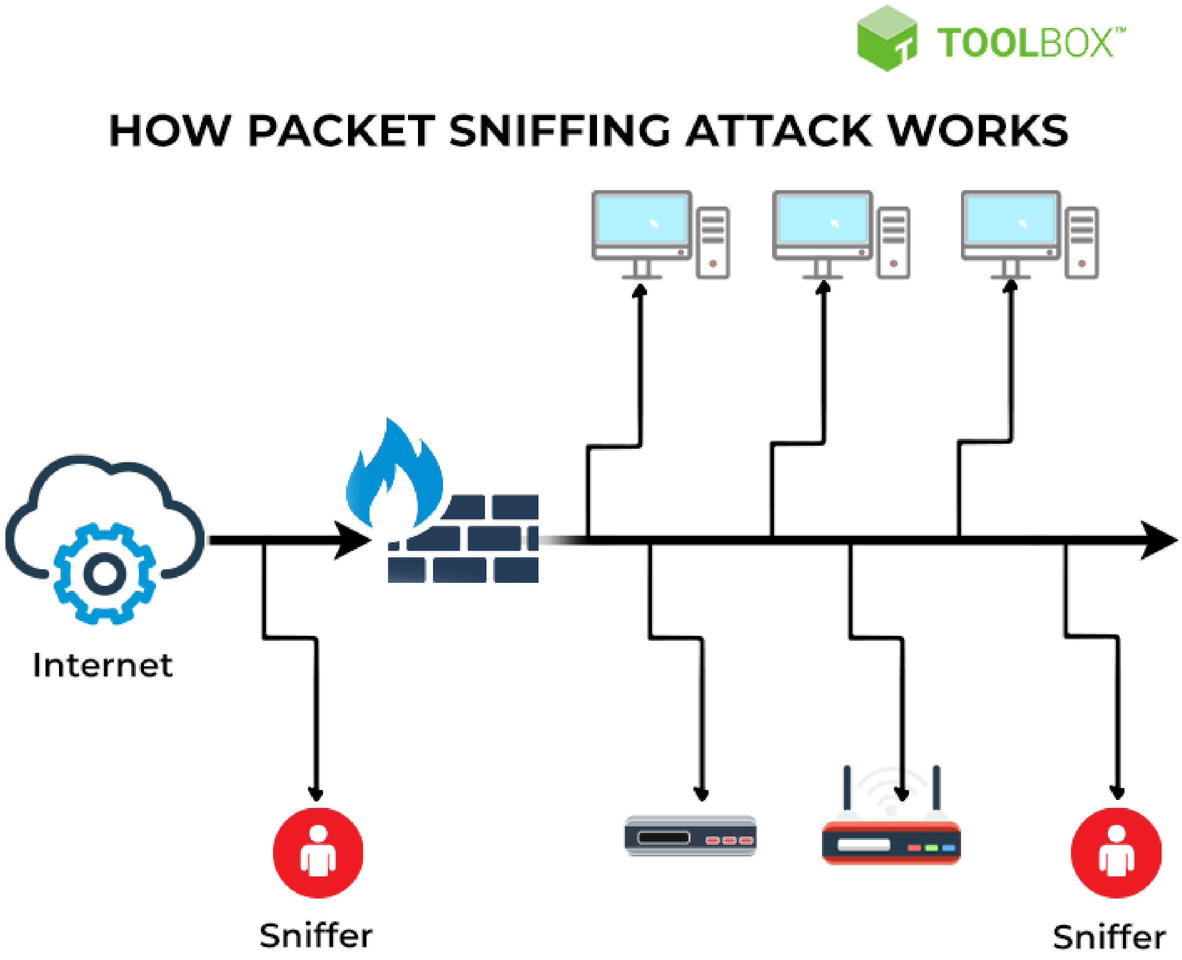
FCS	DATA	TCP / PORTS 443	SRC IP 192.168.0.10	DST IP 1.1.1.1	SRC MAC AA:AA:AA:AA:AA	DST MAC BB:BB:BB:BB:BB

Life of a Packet

Packet Sniffing

Packet sniffing refers to the process of intercepting and capturing network traffic for analysis. It is widely used in both defensive and offensive security. Tools like Wireshark provide a graphical interface for capturing live packets and decoding protocols, making them ideal for debugging, learning, or forensic investigation. On the other hand, tcpdump is a command-line sniffer that offers filtering capabilities for targeted capture, such as isolating HTTP traffic or DNS requests. Scapy, being a Python-based tool, offers programmable sniffing, allowing researchers to script customized packet captures.

Scapy not only captures packets but can also display them in detail. For example, after sniffing a packet, commands like pkt.show() provide a breakdown of all headers and fields, which is useful for analyzing the structure of network protocols. This makes Scapy a valuable tool in hands-on security labs where precise packet analysis is required.

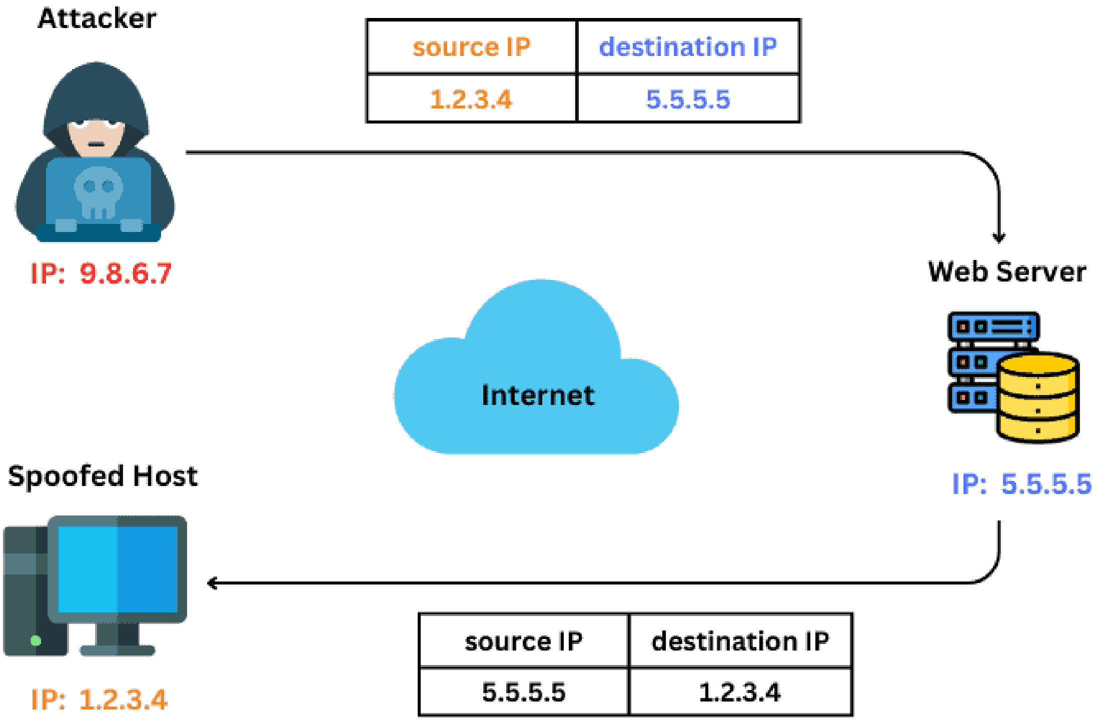


Packet Sniffing

Packet Spoofing

Packet spoofing is the act of forging packets with falsified information, usually by manipulating headers such as IP addresses. A common type is ICMP spoofing, where attackers forge ICMP echo requests or replies with fake source addresses. This technique has been used in classic attacks like the Smurf attack, which floods victims with reflected ICMP responses. Similarly, UDP spoofing involves sending forged UDP packets with fake source addresses, often leveraged in reflection and amplification attacks against servers.

A particularly powerful technique involves sniffing legitimate packets and then spoofing forged responses. By first observing traffic on the network, attackers can craft packets that appear authentic, enabling session hijacking or DNS cache poisoning. This combination of sniffing and spoofing is at the heart of many man-in-the-middle (MITM) attacks.



Packet Spoofing

More About Scapy

Scapy provides deep control over TCP/IP packets through its Python classes. Each protocol layer (IP, TCP, UDP, ICMP, etc.) is represented as a class that can be instantiated and modified. For example, IP() can be used to create an IP header, while TCP() creates a TCP header. These classes can be stacked together to construct complete packets. Extracting information from layers is also straightforward; one can use commands like pkt[IP] to obtain the IP layer, or pkt[TCP].flags to inspect the TCP flag settings.

Beyond packet sniffing and spoofing, Scapy has several other uses. It can perform traceroute operations to map the path of packets across the internet, execute port scanning for discovering open services, carry out ARP spoofing to manipulate local traffic, and even fuzz protocols by generating random malformed packets. These capabilities make Scapy a versatile tool for both attackers and defenders in network security.

Containers and Networks

Modern network security experiments often take place within containers, such as those managed by Docker. Containers allow isolated environments where different network scenarios can be simulated safely. Docker Compose is a tool that simplifies the setup of multi-container applications by defining services, networks, and configurations in a YAML file. This enables quick deployment of realistic lab environments for practicing attacks and defenses.

Docker also provides options for setting up virtual networks. Containers can be attached to bridge networks, giving each one its own IP address within the virtual subnet. This setup allows containers to communicate with one another as though they were on a real network. When setting up containers, each is equipped with a virtual NIC, and security tools or services can be run inside them, ranging from web servers to DNS resolvers.

A key benefit of containers in security education is the ability to sniff traffic inside them. Tools like Wireshark, tcpdump, or

Scapy can be run within containers to analyze traffic between services. This provides a controlled and safe environment for experimenting with sniffing, spoofing, and other network attacks, all without affecting the host system or external networks.

2. THE MAC LAYER AND ATTACKS

The Media Access Control (MAC) layer is a critical component of the data link layer in the OSI model. It is responsible for ensuring reliable communication between devices on the same local area network (LAN). The MAC layer governs how network interface cards (NICs) access the physical medium and transmit frames. Security at this layer is fundamental because many network attacks exploit weaknesses in how MAC addresses are assigned, transmitted, and verified. Since all devices on a LAN can potentially “hear” broadcast traffic, attackers can exploit this visibility to perform sniffing, spoofing, and man-in-the-middle attacks.

Network Interface Card (NIC)

A Network Interface Card (NIC) is the hardware that connects a device to a network. Every NIC has a unique identifier called a MAC address, which is burned into the card by the manufacturer. The NIC handles the conversion of digital data into signals suitable for transmission over the physical medium, and vice versa. It can operate in normal mode, where it only processes packets addressed to it, or in promiscuous mode, where it captures all packets passing through the LAN—making it useful for sniffing attacks or network monitoring.

- **MAC Address**

A MAC (Media Access Control) address is a 48-bit hardware address uniquely identifying each NIC. It is typically expressed in hexadecimal (e.g., 00:1A:2B:3C:4D:5E). The first 24 bits represent the Organizationally Unique Identifier (OUI) assigned to the manufacturer, while the last 24 bits are a unique device identifier. Since MAC addresses are globally unique, they are widely used for device identification. However, MAC addresses can be spoofed, enabling attackers to impersonate another device on the LAN.

- **Virtual Network Interface**

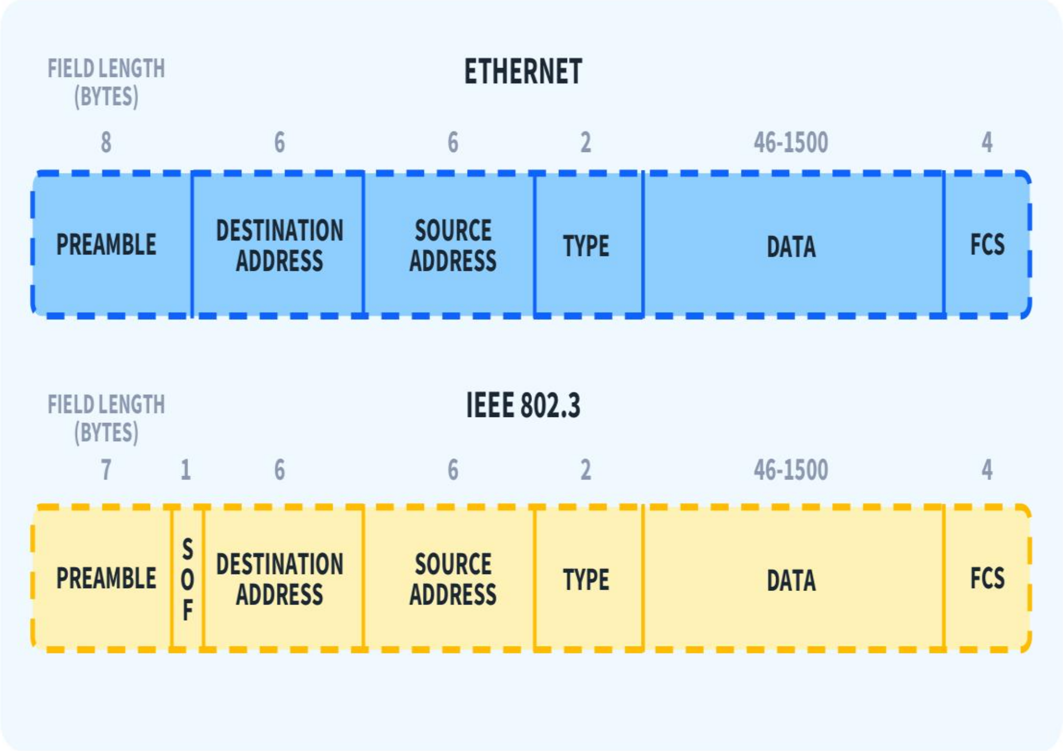
A virtual NIC is a software-emulated network card that operates like a physical NIC but exists only in software. Virtual NICs are common in virtual machines, containers, and VPNs. They allow isolated environments to communicate over networks while still appearing as independent devices with their own MAC addresses. Attackers can exploit virtual NICs for stealthy operations, and defenders can use them to build secure sandboxed environments.

Ethernet Frame

The Ethernet frame is the fundamental unit of data transmission at the MAC layer. It consists of key fields:

- Destination MAC address (6 bytes) – identifies the recipient.
- Source MAC address (6 bytes) – identifies the sender.
- EtherType (2 bytes) – indicates the protocol (e.g., IPv4, IPv6, ARP).
- Payload/Data (46–1500 bytes) – carries the actual data.
- Frame Check Sequence (4 bytes) – used for error detection.

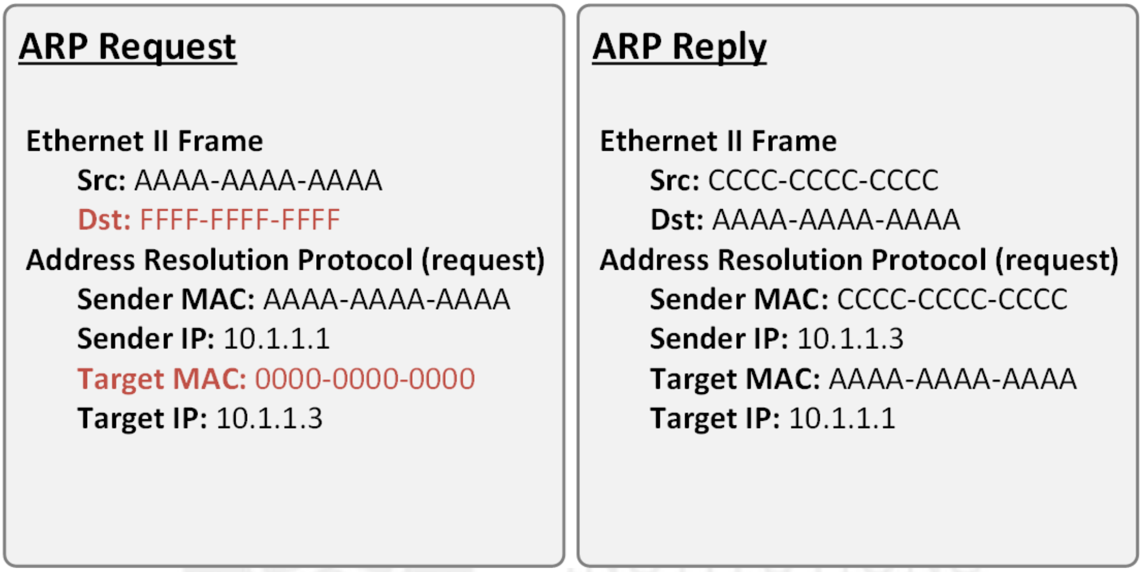
Attackers often manipulate Ethernet frames to perform spoofing or inject malicious traffic into the LAN.



Ethernet Frame

ARP (Address Resolution Protocol)

The ARP protocol is responsible for mapping IP addresses to MAC addresses within a local network. When a device wants to send a packet to another device, it first checks its ARP cache. If no entry exists, it broadcasts an ARP request asking, “Who has this IP?” The owner replies with its MAC address.



ARP Request

Ethernet II Frame
  Src: AAAA-AAAA-AAAA
  Dst: FFFF-FFFF-FFFF
Address Resolution Protocol (request)
  Sender MAC: AAAA-AAAA-AAAA
  Sender IP: 10.1.1.1
  Target MAC: 0000-0000-0000
  Target IP: 10.1.1.3

ARP Reply

Ethernet II Frame
  Src: CCCC-CCCC-CCCC
  Dst: AAAA-AAAA-AAAA
Address Resolution Protocol (request)
  Sender MAC: CCCC-CCCC-CCCC
  Sender IP: 10.1.1.3
  Target MAC: AAAA-AAAA-AAAA
  Target IP: 10.1.1.1

ARP Request and Reply

- **ARP Message Format**

An ARP message contains:

- Sender’s MAC and IP address
- Target’s MAC and IP address These messages can be easily forged since ARP lacks authentication.

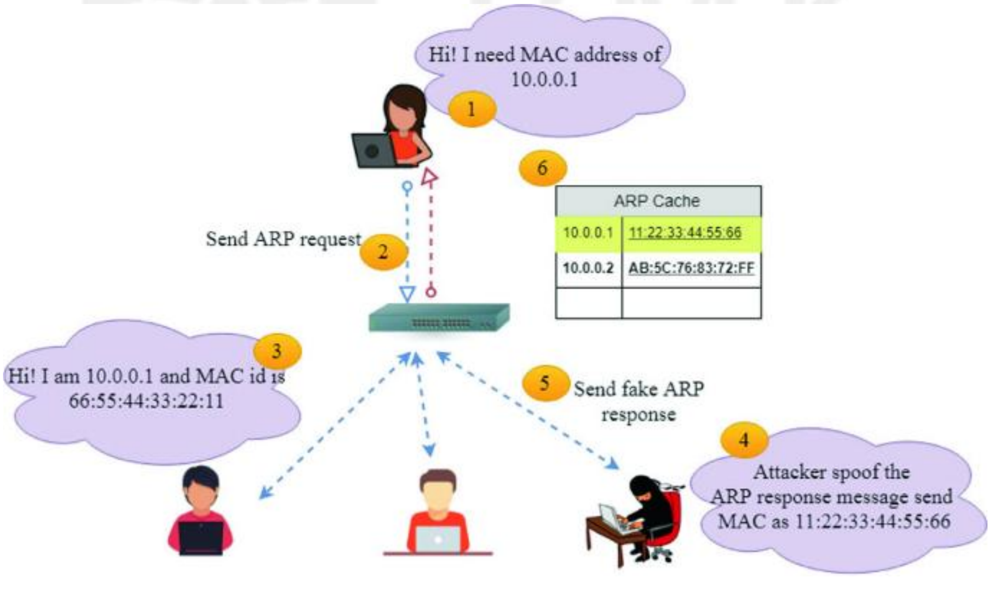
- ARP Cache

Each device maintains an ARP cache that stores mappings of IP addresses to MAC addresses. This speeds up communication by avoiding repeated ARP requests. However, the ARP cache is vulnerable to poisoning attacks, where attackers insert false entries to redirect traffic.

ARP Cache Poisoning Attack

- **Attack Description**

In ARP cache poisoning, an attacker forges ARP replies with false MAC-IP mappings. For example, the attacker can send a forged ARP reply telling the victim that the router's IP corresponds to the attacker's MAC. As a result, the victim sends all its traffic through the attacker.


Hi! I need MAC address of 10.0.0.1
1
6
ARP Cache
10.0.0.1 11:22:33:44:55:66
10.0.0.2 AB:5C:76:83:72:FF
Send ARP request
2
3
Hi! I am 10.0.0.1 and MAC id is 66:55:44:33:22:11
5
Send fake ARP response
4
Attacker spoof the ARP response message send MAC as 11:22:33:44:55:66

ARP Cache Poisoning Attack

- **Discussions**

ARP cache poisoning is dangerous because it allows attackers to intercept, modify, or block communications. It is the basis for man-in-the-middle (MITM) attacks, session hijacking, and denial-of-service (DoS). Since ARP lacks authentication, these attacks are easy to perform on unprotected LANs.

Man-In-The-Middle Attack Using ARP Cache Poisoning

- **Launching the Attack**

In a MITM attack, the attacker poisons the ARP cache of two victims (e.g., a client and a router). Both parties mistakenly believe the attacker's MAC address corresponds to the other's IP. All traffic is then routed through the attacker.

- **IP Forwarding**

To remain undetected, the attacker enables IP forwarding on their machine. This way, intercepted packets are relayed to the actual destination after inspection or modification, so communication continues without interruption.

- **Modifying Telnet Data**

If the victim uses insecure protocols like Telnet (which transmits credentials in plaintext), the attacker can capture usernames and passwords or even alter transmitted commands. This demonstrates how weak protocols combined with ARP poisoning can lead to severe compromises.

3. THE INTERNET PROTOCOL (IP) AND ATTACKS

The Internet Protocol (IP) is a fundamental part of the TCP/IP protocol suite and serves as the main mechanism for addressing and routing packets across networks. It operates at the network layer (Layer 3) of the OSI model and provides a connectionless, best-effort delivery system. This means that IP does not guarantee delivery, ordering, or protection from duplication — it simply makes its best attempt to deliver packets. Reliability and sequencing are delegated to higher-layer protocols like TCP.

Despite its strengths in scalability and simplicity, IP was designed in an era when security was not a priority. As a result, it lacks authentication, encryption, and integrity checks, making it highly vulnerable to a variety of attacks. Understanding IP and its weaknesses is crucial for analyzing network threats.

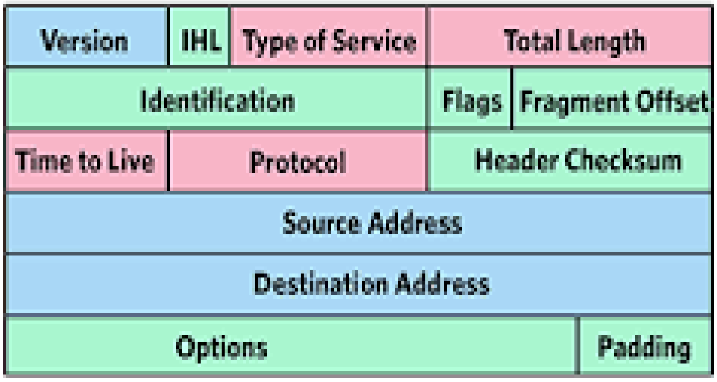
IP Header

Every IP packet begins with a header that contains essential information for delivery and routing. The IPv4 header includes the following important fields:

- Version – Indicates whether the packet is IPv4 or IPv6.
- Header Length (IHL) – Specifies the length of the header, since options may extend it beyond the minimum 20 bytes.
- Type of Service (ToS)/Differentiated Services – Used for prioritization and quality of service.
- Total Length – Size of the entire packet (header + data).
- Identification, Flags, and Fragment Offset – Used to support fragmentation and reassembly.

- Time-To-Live (TTL) – Ensures packets are not forwarded indefinitely.
- Protocol – Indicates the higher-level protocol (e.g., TCP, UDP, ICMP).
- Header Checksum – Provides error detection for the header itself.
- Source and Destination IP addresses – Critical for routing the packet across the network.

Attackers frequently manipulate header fields for spoofing, evasion, and reconnaissance purposes. For instance, changing source addresses enables attackers to hide their identity or launch reflection/amplification attacks.



<table>
<tr><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="3">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td colspan="2">Fragment Offset</td></tr>
<tr><td>Time to Live</td><td colspan="2">Protocol</td><td colspan="3">Header Checksum</td></tr>
<tr><td colspan="6">Source Address</td></tr>
<tr><td colspan="6">Destination Address</td></tr>
<tr><td colspan="5">Options</td><td>Padding</td></tr>
</table>

IP Header

- **Time-To-Live (TTL) and Traceroute**

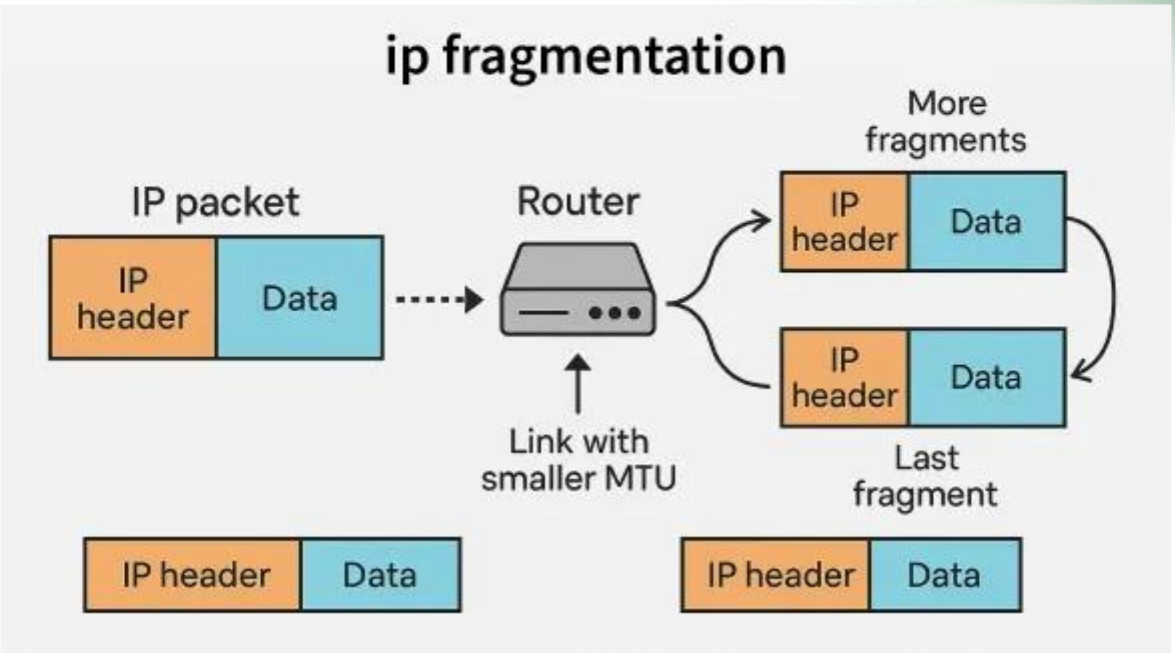
The TTL field prevents packets from looping endlessly in case of routing issues. Each router that handles the packet decrements the TTL by one; when it reaches zero, the router discards the packet and sends an ICMP Time Exceeded message back to the sender.

This mechanism is used by the diagnostic tool traceroute, which maps the path a packet takes through the network. Traceroute sends packets with incrementally increasing TTL values. The first packet (TTL=1) is discarded by the first router, which replies with an error message. The second packet (TTL=2) reaches the second router before being dropped, and so on. By recording the source addresses of these error messages, traceroute builds a hop-by-hop map of the route.

While invaluable for network troubleshooting, traceroute also provides attackers with reconnaissance data — revealing router IP addresses, network topologies, and latency details that can be used in planning attacks.

IP Fragmentation and Attacks

When a packet is larger than the Maximum Transmission Unit (MTU) of a network link, it must be fragmented into smaller packets. Each fragment contains its own IP header and carries information that allows the receiving host to reassemble the original packet.



IP Fragmentation

However, fragmentation introduces vulnerabilities:

- Overlapping fragments – Attackers send fragments that overlap when reassembled, potentially bypassing intrusion detection systems (IDS) or causing reassembly errors.
- Tiny fragment attacks – Splitting headers across fragments to evade firewall rules that check packet headers.
- Teardrop attack – Sending fragments with inconsistent offsets, causing reassembly logic to crash or loop.
- Resource exhaustion – Sending a flood of fragments to overwhelm memory and CPU resources required for reassembly, leading to denial-of-service (DoS).

These attacks exploit the complexity of fragmentation and reassembly, highlighting how features meant for flexibility can be turned into attack vectors.

Routing

Routing is the process by which packets are forwarded from one network to another until they reach their destination. End hosts typically have simple routing tables with a default gateway, while routers maintain more complex tables that define optimal paths based on routing protocols such as RIP, OSPF, or BGP.

Attackers can exploit routing in several ways:

- Route injection – Introducing false routes into a routing table to redirect traffic
- Prefix hijacking – Advertising ownership of an IP prefix to reroute large volumes of traffic.
- Source address spoofing – Sending packets with fake source addresses, making attacks difficult to trace.

A common defense is Reverse Path Filtering (RPF), which ensures that incoming packets have a valid route back to their claimed source. If the source address is not reachable via the incoming interface, the packet is dropped, thereby reducing spoofed traffic.

ICMP and Attacks

The Internet Control Message Protocol (ICMP) is used by IP to send control, error, and diagnostic messages. Common uses include ping, which tests connectivity using ICMP Echo Request and Echo Reply messages.

Despite its utility, ICMP has been abused in numerous attacks:

- Smurf Attack – An attacker sends ICMP Echo Requests with the victim’s IP address as the spoofed source to broadcast addresses. All devices on the broadcast network reply to the victim, amplifying the attack into a powerful DoS.
- ICMP Redirect Attack – Normally, routers use ICMP redirect messages to inform hosts of better routes. An attacker can forge these messages to trick a host into routing traffic through the attacker’s machine, enabling a Man-in-the-Middle (MITM) attack.
- MITM with ICMP Redirect – After redirecting traffic, attackers can intercept, modify, or drop packets, compromising confidentiality and integrity.
- Other ICMP-based attacks – ICMP tunneling (using ICMP packets to smuggle hidden data), reconnaissance (mapping networks by probing), and resource exhaustion attacks.

Because of these risks, many networks restrict or carefully monitor ICMP traffic.

Network Address Translation (NAT)

Network Address Translation (NAT) is a technique used to map multiple private IP addresses within a local network to a single public IP address for external communication. This is particularly useful in IPv4 networks facing address scarcity.

NAT works by maintaining a translation table that records which private address and port map to which public address and port. Outgoing packets have their private addresses replaced with the public address, while incoming packets are translated back. Variants include Static NAT, Dynamic NAT, and Port Address Translation (PAT).

From a security perspective, NAT provides a degree of anonymity by hiding internal structures from outsiders. However, it is not a true security mechanism. Attackers may attempt to exploit misconfigured NAT devices, bypass firewall policies, or use NAT to conceal malicious activity. NAT can also complicate security monitoring and end-to-end encryption.

4. PACKET SNIFFING AND SPOOFING

Introduction

Packet sniffing and spoofing are two fundamental techniques in network security and attack scenarios.

- Packet sniffing is the process of intercepting and analyzing network traffic. Security professionals use it for troubleshooting, monitoring, and intrusion detection, while attackers use it to steal sensitive information such as passwords, cookies, or session data.

- Packet spoofing involves forging packets with false headers (e.g., fake source addresses). This can hide the attacker’s identity, impersonate other machines, or redirect traffic in malicious ways.

Together, sniffing and spoofing form the basis of many advanced attacks such as Man-in-the-Middle (MITM), session hijacking, and DoS amplification attacks.

How Packets Are Received

Network Interface Card (NIC)

The Network Interface Card (NIC) is the hardware device that connects a computer to a network. Normally, the NIC operates in non-promiscuous mode, meaning it only processes frames addressed to its own MAC address, plus broadcast/multicast packets.

When placed in promiscuous mode, the NIC forwards all packets it sees on the wire (or wireless medium) to the operating system, regardless of the destination MAC address. Packet sniffers rely on this mode to capture traffic not intended for the host.

BSD Packet Filter (BPF)

The BSD Packet Filter is a kernel-level mechanism that provides efficient packet capturing. It acts as a programmable filter that allows applications (like Wireshark or tcpdump) to specify what packets should be captured. This avoids overwhelming applications with irrelevant packets.

For example, BPF rules can specify:

- Only capture TCP packets on port 80 (HTTP traffic).
- Only capture packets from a specific IP address.
- Drop all others.

This improves performance and makes packet sniffing scalable.

Packet Sniffing

- **Receiving Packets Using Sockets**

Sockets are programming interfaces that allow communication between applications and the network stack. Normal sockets (e.g., TCP or UDP sockets) only allow access to payloads after the OS has processed headers. For sniffing, raw sockets are used. Raw sockets provide access to the entire packet, including headers.

- **Packet Sniffing Using Raw Sockets**

Raw sockets allow applications to bypass the transport layer and directly capture packets at the IP layer. However, raw sockets often require administrator privileges and are restricted in modern systems due to security concerns.

Limitations:

- Not portable across operating systems.
- Cannot easily apply filters like BPF.
- Higher overhead for large traffic volumes.

Packet Sniffing Using the pcap API

The pcap (packet capture) API is a standardized library used in tools like tcpdump and Wireshark. It provides:

- High performance capturing.
- BPF-based filtering.
- Portability across Unix/Linux/Windows.

libpcap (C library) and WinPcap (Windows variant) are widely used. Applications capture packets via pcap functions and process them for logging, analysis, or visualization.

- **Processing Captured Packets**

After capturing, packets must be parsed to extract meaningful information. This involves:

- Decoding Ethernet, IP, TCP/UDP, and application layer headers.
- Reassembling fragmented packets.
- Extracting credentials or payloads (e.g., HTTP requests, Telnet logins).

Attackers often automate processing to detect useful data in real time, while defenders use it for intrusion detection.

•Packet Sniffing Using Scapy

Scapy is a powerful Python-based tool for packet crafting, sniffing, and manipulation. Unlike tcpdump (which is passive), Scapy allows interactive sniffing and modification. For example:

- Capturing DNS requests.
- Analyzing TCP handshakes.
- Injecting or replaying packets.

It is widely used in penetration testing and research.

•Special Notes Regarding Containers

In containerized environments (like Docker or Kubernetes), packet sniffing is trickier because containers use virtual network interfaces and overlays. Sniffers must often run with elevated privileges or inside the host namespace. Attackers can exploit this to escape monitoring, while defenders must carefully configure monitoring tools.

Packet Spoofing

•Sending Normal Packets Using Socket

Applications can use sockets to send normal packets, typically via transport protocols (TCP/UDP). In this case, the OS handles packet headers automatically.

•Sending Spoofed Packets Using Raw Sockets

To spoof packets, attackers use raw sockets that allow manual construction of headers. For example, they can:

- Set a fake source IP.
- Manipulate TCP flags to bypass firewalls.
- Craft packets that look like they come from trusted hosts.

Modern systems restrict raw sockets, but attackers bypass this using tools like Scapy or packet injection frameworks.

Constructing ICMP Packets

ICMP spoofing involves forging packets such as Echo Requests (ping) or Echo Replies. Attackers use this for:

- DoS amplification (Smurf attack).
- Redirecting traffic (ICMP Redirect attack).
- Network scanning (stealth pings).

Constructing UDP Packets

UDP is connectionless, making spoofing easy. Attackers can forge UDP packets for:

- Amplification attacks (e.g., DNS amplification, NTP amplification).
- Bypassing firewalls.
- Flooding attacks against services.

Sniffing and Then Spoofing

A powerful attack strategy involves combining sniffing with spoofing. The attacker first sniffs traffic to gather information.

about active connections (e.g., sequence numbers in TCP). Then, they inject forged packets to:

- Hijack sessions.
- Reset connections.
- Modify data in transit.

This is the basis of Man-in-the-Middle and session hijacking attacks.

Spoofing Packets Using a Hybrid Approach

- **A Hybrid Approach**

Instead of manually constructing packets from scratch, attackers often mix automated tools with custom code. For example, using Scapy to define a packet template, then using low-level C code to rapidly inject large volumes of spoofed packets.

- **Constructing Packet Template Using Scapy**

Scapy can generate a packet skeleton (Ethernet, IP, TCP/UDP/ICMP headers). Attackers then export this template.

- **Modifying and Sending Packets Using C**

C programs using raw sockets or libraries can modify fields (e.g., IP addresses, ports) and send packets at high speed. This makes large-scale spoofing more efficient than Python-only approaches.

Endianness

Endianness refers to the byte order in which multi-byte values are stored:

- Big-endian – most significant byte first (used in network protocols, also called network byte order).
- Little-endian – least significant byte first (used by Intel CPUs).

When constructing or parsing packets, programmers must ensure correct endianness conversions, or the packet will be invalid.

- **Calculating Checksum**

Checksums are used in IP, TCP, and UDP headers to verify data integrity. When spoofing packets, attackers must correctly calculate and insert checksums, otherwise routers or endpoints will discard the packets.

Checksums are computed using one’s complement addition of 16-bit words in the header and payload. Libraries like Scapy automate this, but attackers writing raw code must implement it manually.

Summary

Packet sniffing and spoofing are core concepts in both network security defense and cyber attacks.

- Sniffing allows monitoring and extraction of sensitive information.

- Spoofing enables attackers to impersonate others, hide identities, and launch advanced attacks.
- Together, they underpin techniques like MITM, DoS amplification, and session hijacking.

Understanding these mechanisms is essential for building secure networks and detecting malicious behavior.

5. ATTACKS ON TCP PROTOCOL

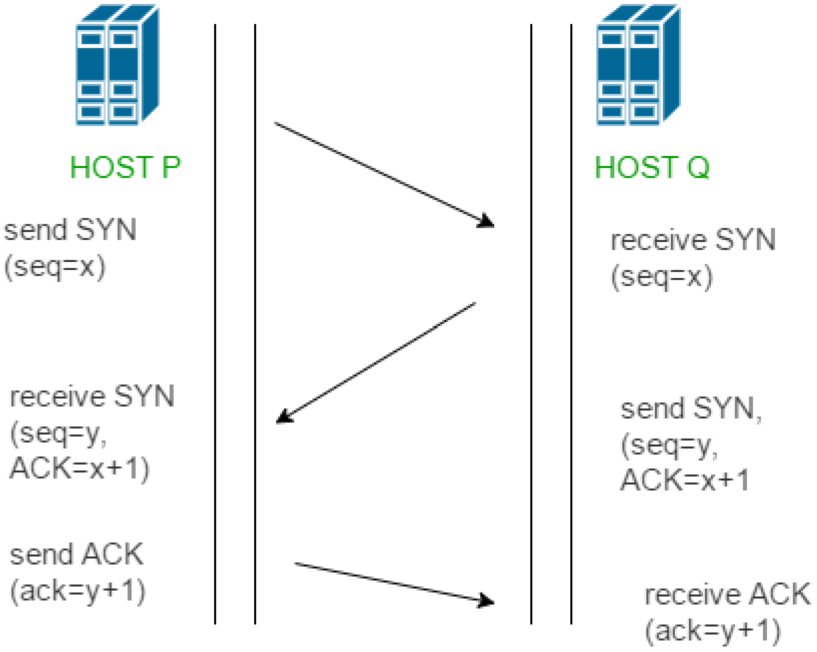
The Transmission Control Protocol (TCP) is one of the most widely used transport-layer protocols, providing reliable, connection-oriented communication between applications. TCP ensures data integrity and correct sequencing through mechanisms such as acknowledgments, retransmissions, and flow control. However, because of its openness and predictable behavior, TCP is also vulnerable to a number of security attacks. These attacks exploit weaknesses in the way TCP connections are established, maintained, and terminated, often resulting in denial-of-service (DoS), session hijacking, or unauthorized access. Understanding TCP’s structure and its vulnerabilities is essential for defending against malicious exploitation.

How the TCP Protocol Works

TCP works by establishing a virtual connection between a client and server before data transmission. The communication process begins with a three-way handshake, where both sides synchronize and acknowledge each other’s sequence numbers. Once established, data flows in an ordered and reliable fashion, with TCP headers carrying sequence and acknowledgment numbers, source and destination ports, flags, and other fields critical for connection management. While this predictability makes TCP reliable, it also enables attackers to anticipate responses, inject malicious packets, or disrupt sessions by spoofing packets. TCP’s design thus becomes a double-edged sword, providing robustness while exposing vulnerabilities.

- **TCP Three-Way Handshake Protocol**

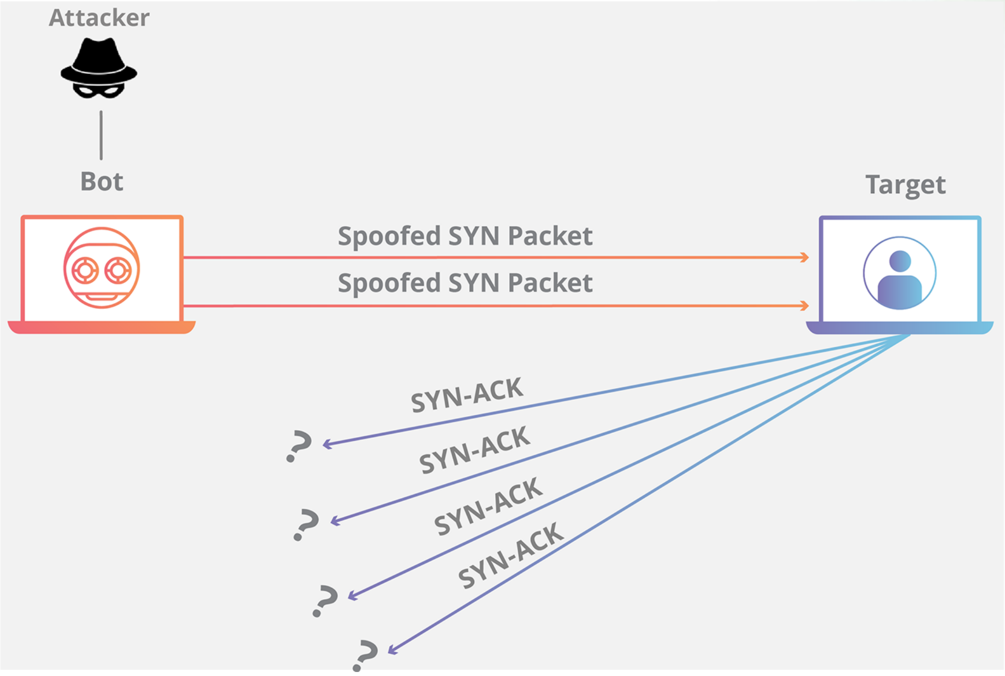
The three-way handshake is the foundation of TCP connection establishment. First, the client sends a SYN (synchronize) packet to the server, initiating the request. The server responds with a SYN+ACK packet, acknowledging the request and providing its own sequence number. Finally, the client sends back an ACK packet, confirming the connection. At this point, the connection is established, and data transmission can begin. Attackers often exploit this predictable pattern by sending large volumes of SYN packets without completing the handshake, overwhelming the server’s resources. This forms the basis of the SYN flooding attack.



TCP 3 way Handshake

The SYN Flooding Attack

SYN flooding is a classic denial-of-service (DoS) attack that targets the three-way handshake mechanism. An attacker sends a large number of SYN requests to a server but deliberately does not complete the handshake. The server allocates memory and resources for each half-open connection, eventually exhausting its capacity to handle legitimate requests. As a result, legitimate users are unable to establish TCP connections. This attack does not require high bandwidth and can be launched with limited resources, making it particularly effective and dangerous.



SYN Flood Attack

- **Launching the SYN Flooding Attack**

In practice, attackers use raw sockets or packet crafting tools such as Scapy to send forged SYN packets. These packets often use spoofed source IP addresses to hide the attacker's identity and make defense more difficult. Because the server waits for acknowledgments that never arrive, the backlog queue fills up quickly, denying service to legitimate clients. Modern systems implement various defenses, but unprotected systems remain vulnerable.

Issues in SYN Flooding Attacks

Although SYN flooding is powerful, it comes with challenges for the attacker. For example, network intrusion detection systems (NIDS) can detect abnormal traffic patterns, and spoofed IP addresses may not always work if ISPs implement ingress filtering. Additionally, the attack requires sustained traffic to keep the server overloaded. Defenders often use SYN cookies or limit the backlog queue to mitigate the impact. Despite these countermeasures, the attack remains a threat, especially when combined with distributed botnets.

Launching SYN Flooding Attacks Using C Code

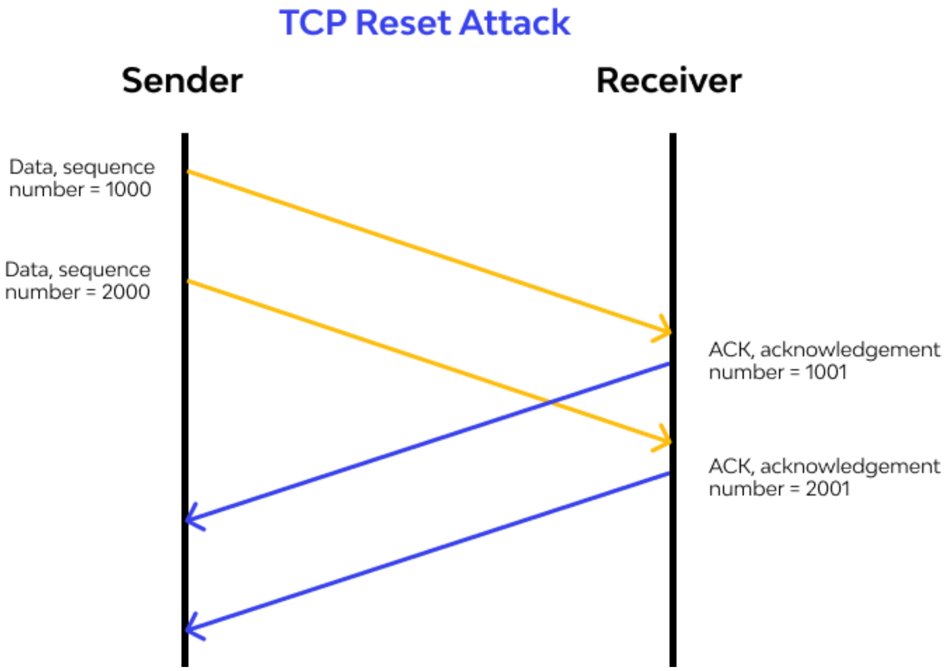
SYN flooding can be programmatically launched by crafting packets with custom TCP headers in low-level programming languages like C. Using raw sockets, attackers can set arbitrary sequence numbers, spoof IP addresses, and send packets directly to the victim. Such attacks bypass the operating system's normal TCP stack, giving attackers fine-grained control over packet construction. While educationally useful for understanding TCP behavior, this technique is also used by malicious actors to perform large-scale denial-of-service attacks.

Countermeasures

Defenses against SYN flooding include SYN cookies, which allow the server to handle large volumes of SYN packets without allocating resources until the handshake is completed. Firewalls and intrusion detection systems can also filter abnormal traffic patterns. Rate limiting, ingress filtering, and increasing backlog queue size are additional protective strategies. However, distributed denial-of-service (DDoS) attacks with SYN flooding remain difficult to fully mitigate, making layered defenses necessary.

TCP Reset Attack

TCP connections can be forcefully terminated by sending forged TCP reset (RST) packets. In a TCP reset attack, the attacker crafts a packet with the RST flag set and spoofs it to appear as if it came from one of the legitimate endpoints of a TCP connection. Since TCP accepts RST packets as valid instructions to close connections, the communication is abruptly terminated. This attack is particularly disruptive against long-lived connections such as Telnet, SSH, or video streams.


TCP Reset Attack
Sender
Receiver
Data, sequence number = 1000
Data, sequence number = 2000
ACK, acknowledgement number = 1001
ACK, acknowledgement number = 2001

TCP Reset Attack on Different Connections

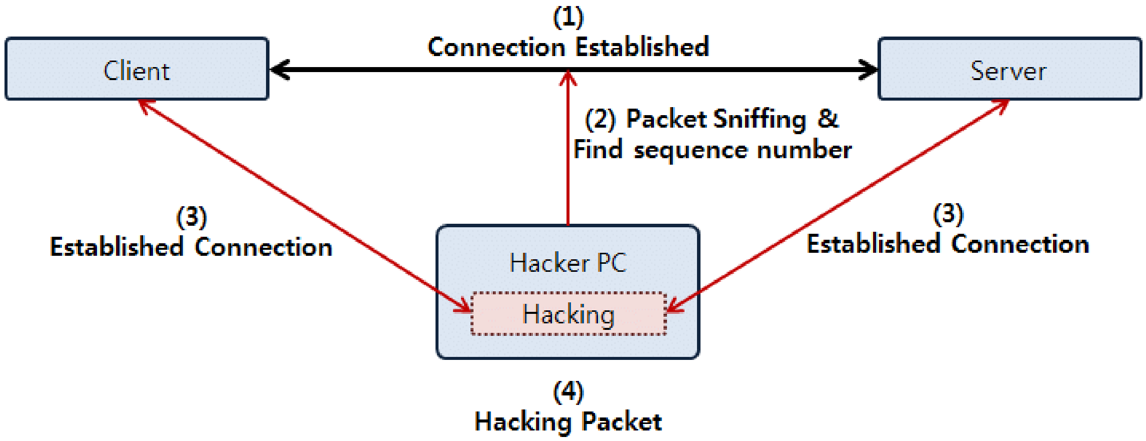
Telnet connections are highly vulnerable since they transmit data in plaintext and are interactive by nature. Injecting a forged RST packet can immediately disconnect the user

SSH connections, though encrypted, can still be reset by forged RST packets because the reset occurs at the TCP level before encryption is relevant.

Video-streaming connections are also disrupted easily, leading to degraded service quality or forced disconnections.

TCP Session Hijacking Attack

Session hijacking occurs when an attacker takes over an active TCP connection between two parties. By predicting or sniffing the sequence numbers in a session, the attacker can inject malicious packets and impersonate one side of the communication. Once hijacked, the attacker can issue commands, steal data, or disrupt communication. This is particularly dangerous in remote login services, where an attacker could gain unauthorized access to systems by hijacking administrative sessions.



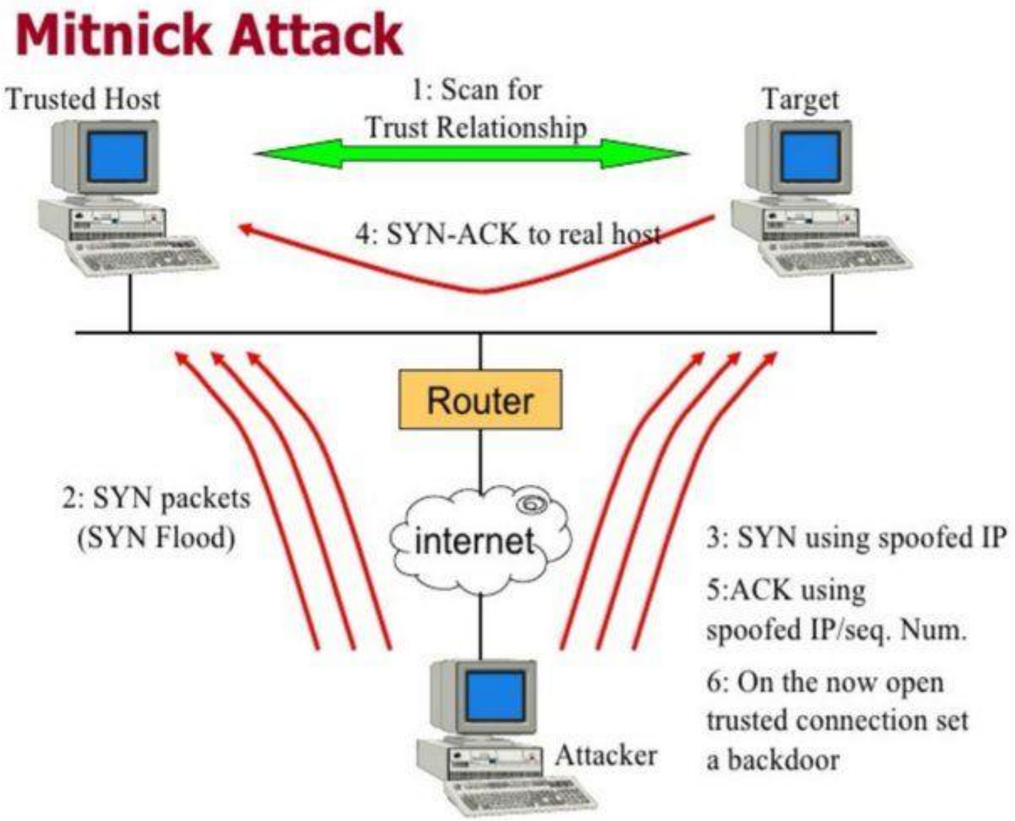
TCP Session Hijacking Attack

Causing More Damage and Reverse Shells

After hijacking a TCP session, attackers can escalate their control by injecting harmful payloads. For example, they can upload a reverse shell, allowing them persistent control of the victim's machine. In this scenario, the victim's system initiates a connection back to the attacker, bypassing firewall restrictions. Such capabilities highlight how TCP session hijacking can evolve from temporary disruption into full system compromise.

The Mitnick Attack

The Mitnick Attack, named after hacker Kevin Mitnick, is one of the most famous examples of TCP spoofing. The attacker first silences the trusted server (for example, by launching a DoS attack against it) and then forges a spoofed TCP connection with a target machine by predicting sequence numbers. Once the spoofed connection is established, the attacker gains access to the target under the guise of the trusted host. This attack demonstrated the practical dangers of TCP sequence number prediction and highlighted the need for stronger authentication mechanisms beyond IP and TCP trust relationships.


Mitnick Attack
Trusted Host
1: Scan for Trust Relationship
Target
4: SYN-ACK to real host
Router
2: SYN packets (SYN Flood)
internet
3: SYN using spoofed IP
5:ACK using spoofed IP/seq. Num.
6: On the now open trusted connection set a backdoor
Attacker

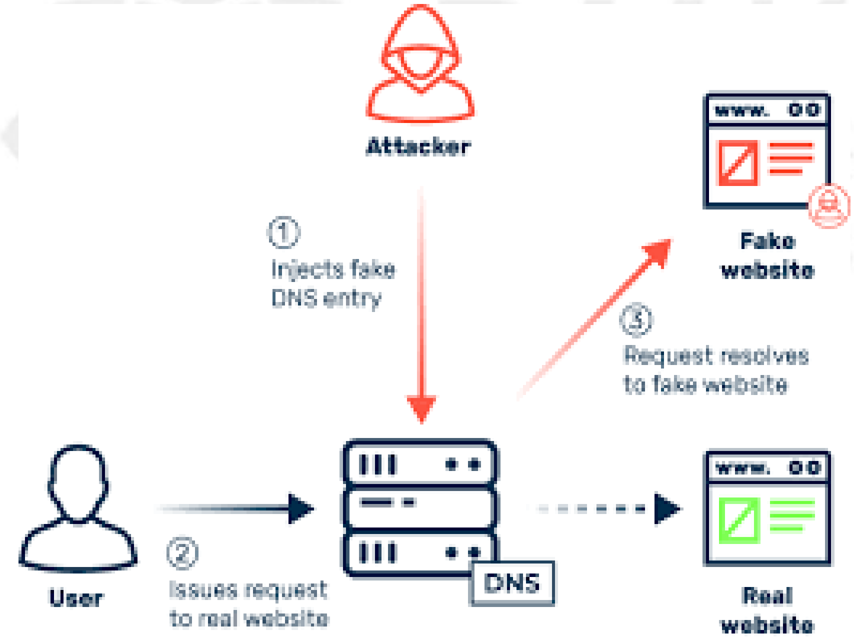
Mitnick Attack

6. DNS ATTACKS: OVERVIEW

DNS attacks exploit vulnerabilities in the Domain Name System, a core part of the internet responsible for mapping human-readable domain names to IP addresses. Since DNS operates on trust and was originally designed without strong authentication, attackers can manipulate responses, inject malicious records, or overwhelm servers. These attacks can lead to traffic redirection, data theft, denial of service, or complete compromise of user trust in web services. Understanding different forms of DNS attacks is essential for deploying effective countermeasures.

7. LOCAL DNS CACHE POISONING ATTACK

In local DNS cache poisoning, an attacker injects false DNS records into the cache of a DNS resolver within a local network. This misleads users into being redirected to malicious servers instead of legitimate ones. The attacker typically sends forged responses faster than the real server’s reply, tricking the resolver into caching malicious entries. Once poisoned, all subsequent requests for the targeted domain resolve incorrectly until the cache expires, making this attack highly effective for phishing and malware delivery.



DNS Cache Poisoning Attack

- ### Launch DNS Cache Poisoning Attack

Launching such an attack requires the adversary to predict transaction IDs or manipulate weakly secured resolvers. By flooding the local DNS resolver with fake responses that appear authentic, attackers aim to overwrite legitimate records with harmful ones. The attacker’s goal is to ensure their malicious DNS reply matches the query before the actual authoritative server responds, granting them control over how users are redirected.

- **Targeting the Authority Section**

Instead of directly corrupting the answer section, attackers may insert fake data into the authority section of DNS responses. By doing this, they trick resolvers into querying malicious authoritative servers for further records. This indirect poisoning technique is more subtle and can extend the scope of manipulation to multiple domains managed by the fake authority.

8. REMOTE DNS CACHE POISONING ATTACK

Remote cache poisoning attacks target resolvers beyond the attacker’s immediate network. Since predicting transaction IDs and source ports is difficult, attackers exploit weaknesses such as insufficient entropy in request generation. By sending numerous forged DNS replies, they aim to inject malicious records into the cache of remote resolvers, which then mislead a wide range of users.

- **The Kaminsky Attack**

One of the most famous DNS cache poisoning methods is the Kaminsky attack. It exploits the fact that DNS responses can contain additional information besides the requested record.

By triggering multiple queries for random subdomains of a target, the attacker floods the resolver with spoofed responses. If one matches correctly, the attacker poisons the cache with fake entries, allowing them to control large swaths of domain resolution for extended periods.

▪ Constructing the IP and UDP Headers of DNS Reply

To carry out poisoning attacks effectively, attackers forge custom DNS packets. This involves constructing fake IP headers and UDP headers that appear to come from legitimate authoritative servers. Proper spoofing ensures that the fake packet passes basic validation checks, thereby making the attack more believable to the resolver.

▪ Constructing the DNS Header and Payload

Beyond headers, attackers carefully craft the DNS payload. This includes setting the transaction ID to match the query, inserting forged answer records, and manipulating authority and additional sections. The crafted payload makes the fake response indistinguishable from a real one, significantly raising the chance of poisoning success.

9. REPLY FORGERY ATTACKS FROM MALICIOUS DNS SERVERS

In this scenario, attackers operate or compromise DNS servers to deliberately send falsified replies. Instead of legitimate records, they return forged IP addresses, redirecting traffic. Since resolvers inherently trust authoritative servers, this type of attack can easily spread malicious data across networks, bypassing traditional defenses.

- **Fake Data in the Additional Section**

A common method of reply forgery is inserting fake IP addresses for name servers in the additional section of DNS responses. This tricks resolvers into contacting malicious servers for further queries, expanding the attacker's control over subsequent lookups.

- **Fake Data in the Authority Section**

By forging authority records, attackers deceive resolvers into trusting malicious servers as authoritative for particular domains. This manipulation enables attackers to answer future queries with complete control.

- **Fake Data in Both Authority and Additional Sections**

A more dangerous variant occurs when attackers inject forged data into both sections simultaneously. This ensures resolvers not only recognize malicious servers as authoritative but also know how to reach them, making the forgery seamless.

- **Fake Data in the Answer Section**

This is the most direct form of reply forgery, where the attacker replaces the legitimate IP address in the answer section with a malicious one. Victims who query that domain are immediately redirected to attacker-controlled infrastructure.

- **Fake Answer in Reverse DNS Lookup**

Attackers may also manipulate reverse DNS lookups, which map IP addresses to domain names. By supplying fake names, they can mislead logging systems, security tools, or administrators into trusting malicious hosts disguised as legitimate ones.

10. DNS REBINDING ATTACK

DNS rebinding attacks exploit the way browsers enforce the Same-Origin Policy. By rapidly changing DNS responses, an attacker tricks a victim's browser into treating a malicious site and a local resource as the same origin. This allows attackers to bypass restrictions and gain access to internal networks or devices such as routers and IoT systems.

- **How DNS Rebinding Attack Works**

The attack begins when a user visits a malicious site. The site's DNS record is set with a very short time-to-live (TTL). The browser first connects to the malicious server, then receives new DNS replies binding the same domain to local IP addresses. This way, the attacker gains access to private networks through the victim's browser.

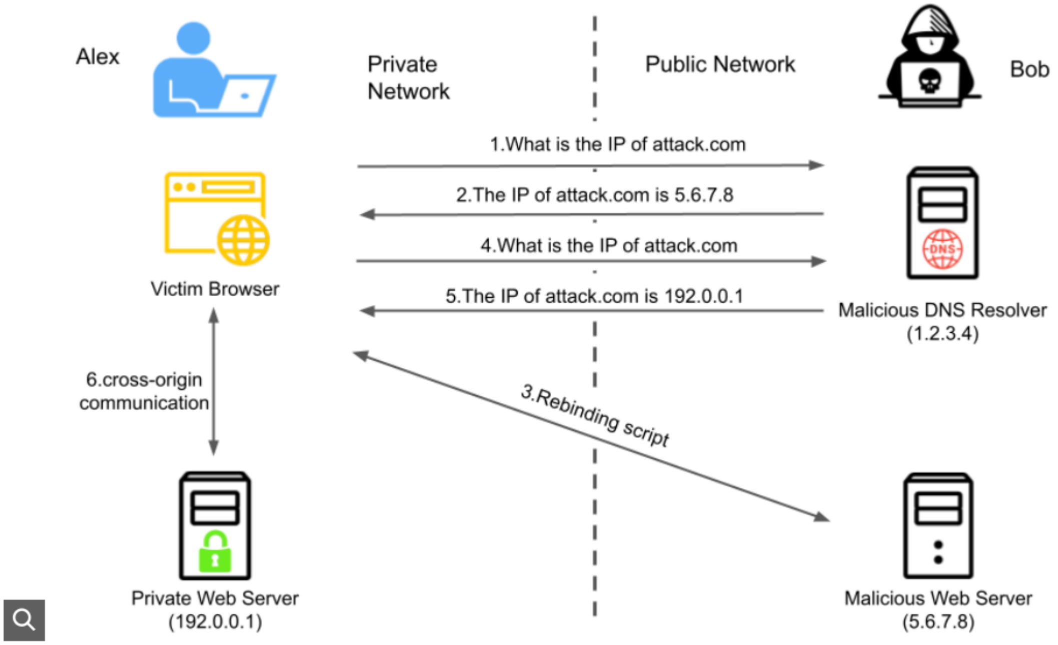

Alex
Private Network
Public Network
Bob
1.What is the IP of attack.com
2.The IP of attack.com is 5.6.7.8
4.What is the IP of attack.com
5.The IP of attack.com is 192.0.0.1
Victim Browser
Malicious DNS Resolver (1.2.3.4)
DNS
6.cross-origin communication
3.Rebinding script
Private Web Server (192.0.0.1)
Malicious Web Server (5.6.7.8)

Figure 1. Mechanism of DNS Rebinding.

11. DENIAL OF SERVICE ATTACKS ON DNS SERVERS

Attackers also target DNS servers with denial-of-service (DoS) attacks. By overwhelming root servers, top-level domain servers, or specific authoritative servers, attackers can cause large-scale disruption. DoS attacks may use amplification techniques, where small queries generate large responses, consuming bandwidth and processing power until servers fail to respond.

- **Attacks on the Root and TLD Servers**

Root and top-level domain (TLD) servers are critical for internet functionality. Targeted attacks on them can disrupt name resolution globally. While these servers are well protected and widely replicated, sustained large-scale attacks can still degrade their performance.

- **Attacks on Nameservers of a Particular Domain**

Attackers may focus on specific domains by targeting their authoritative name servers. This makes the domain inaccessible to legitimate users, causing downtime for businesses, governments, or services.

12. DNSSEC: SECURING DNS

DNS Security Extensions (DNSSEC) were developed to counter cache poisoning and forgery attacks. DNSSEC uses public key cryptography to digitally sign DNS data, ensuring authenticity and integrity. While it does not encrypt queries, it guarantees that responses come from legitimate servers and have not been tampered with during transit.

How DNSSEC Works

DNSSEC introduces digital signatures for DNS records. Each zone signs its data with private keys, and resolvers validate signatures with corresponding public keys. This chain of trust extends from the root zone down to individual domains, making forgery nearly impossible.

Public Keys

The foundation of DNSSEC security lies in public-private key pairs. Public keys are shared openly, allowing resolvers to verify signed records. Without the matching private key, attackers cannot forge valid signatures, preventing tampering.

Generating KSK and ZSK Keys for Zone

DNSSEC distinguishes between Key Signing Keys (KSKs) and Zone Signing Keys (ZSKs). The KSK secures the ZSK, while the ZSK signs the actual zone data. This separation of roles improves security and allows easier key management.

Signing the Zone File

Zone files, which contain DNS records, are digitally signed using the ZSK. These signatures are included in DNS responses, enabling resolvers to verify that data has not been altered.

The DS Record for the Key Signing Key

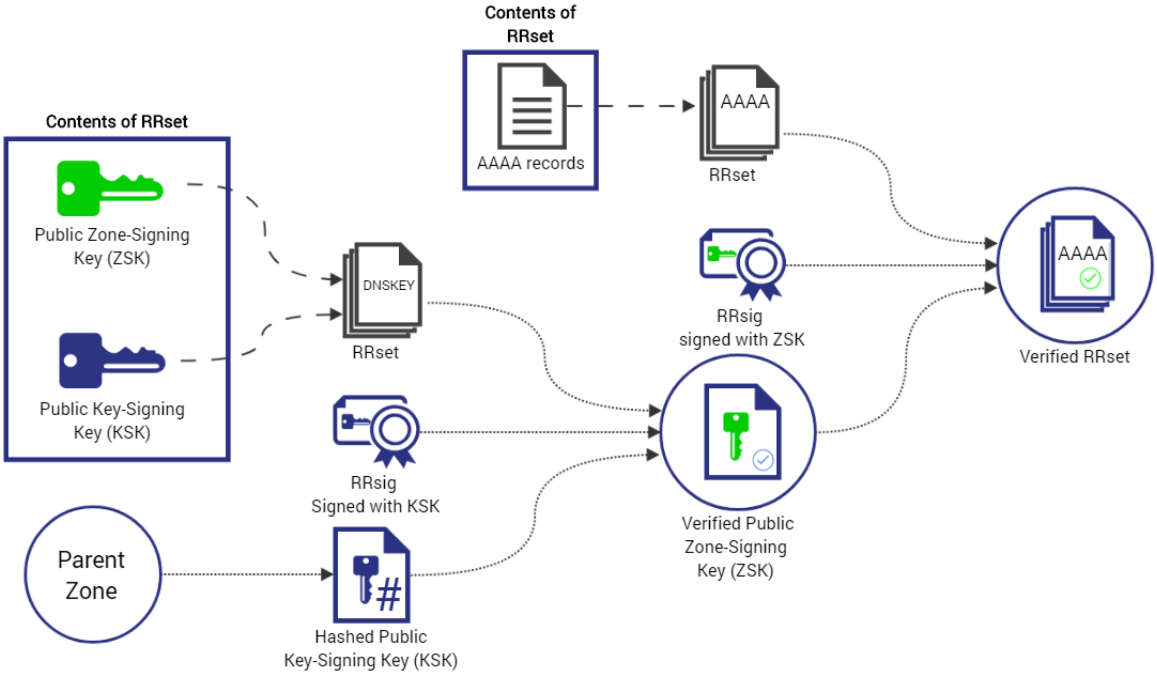
Delegation Signer (DS) records link child zones to parent zones in the chain of trust. The DS record of a domain’s KSK is stored in the parent zone, enabling resolvers to validate authenticity all the way from the root zone.

Hands-on Experience on DNSSEC

Practical deployment of DNSSEC involves setting up signed authoritative servers and validating resolvers. Administrators generate keys, sign zones, and configure servers to support DNSSEC queries. Once active, resolvers can reject tampered or unsigned responses.

TLS/SSL Solution

While DNSSEC ensures authenticity of DNS records, TLS/SSL secures communication between clients and servers. Together, they provide a comprehensive solution: DNSSEC prevents users from being redirected to malicious sites, while TLS/SSL encrypts data exchanged with legitimate servers.



DNSSEC

Lab exercises

Conducting TCP SYN Flood Attack

2.Sniffing Packets on Network Interfaces

3.Spoofing Source IP Address in Packets

4.Investigating DNSSEC Implementation and Validation

Lecture Slides

Lecture Slides
https://drive.google.com/drive/folders/1MdIU8YLJpG_n3byErvlaSDCn4JTDGb23?usp=drive_link

Assignment

Assignment

Q1. Analyze the role of ARP in network communication. Design an experiment using tools like Scapy or Wireshark to demonstrate ARP cache poisoning and explain the security implications. (CO4, K4)

Q2. Implement a packet sniffing program using raw sockets or the pcap API. Capture real-time packets, analyze their headers, and discuss how attackers may misuse this for spoofing. (CO4, K3)

Q3. Conduct a case study on the SYN flooding attack. Using diagrams, explain how the TCP three-way handshake is exploited and evaluate different defense mechanisms such as SYN cookies. (CO4, K5)

Q4. Investigate DNS cache poisoning and DNS rebinding attacks. Compare their techniques, impact on users, and propose practical solutions for mitigating them in modern systems. (CO4, K4)

Q5. Demonstrate the working of DNSSEC in a simulated environment. Explain the process of key generation, zone signing, and validation with examples, and critically analyze its limitations. (CO4, K5)

Part A Q & A

PART -A

Q1. Define network security. (CO4, K1)
Ans: Network security is the practice of protecting data and resources during communication across computer networks. It ensures that sensitive information remains safe from unauthorized access, modification, and theft. The core principles include confidentiality, integrity, and availability of data. It also involves deploying firewalls, intrusion detection systems, and encryption methods to secure communications.

Q2. What are the main goals of network security? (CO4, K1)
Ans: The primary goals are confidentiality, integrity, availability, authentication, and non-repudiation. Confidentiality protects data from unauthorized access, while integrity ensures it is not altered. Availability guarantees services are accessible when needed. Authentication verifies user identity, and non-repudiation prevents denial of actions performed.

Q3. Explain the life cycle of a packet briefly. (CO4, K2)
Ans: A packet's journey begins with construction inside the kernel of a host system. It is then sent through the NIC to the network medium, where routers forward it along the path. At the destination, the receiving NIC and OS process the packet. Finally, the payload is delivered to the application layer.

Q4. What is routing in packet forwarding? (CO4, K2)
Ans: Routing is the mechanism by which packets are directed from a source to a destination across interconnected networks. Routers maintain routing tables and use algorithms to determine the most efficient path. Packets may traverse multiple intermediate nodes before reaching the target. Efficient routing ensures minimal delay and prevents loops.

Q5. Name two packet-sending tools. (CO4, K2)
Ans: Tools like Scapy and tcpdump are commonly used for constructing and sending packets. Scapy allows crafting of custom packets and simulating attacks for research. Tcpdump captures and analyzes live traffic at the packet level. Both are essential for learning packet-level security testing and attacks.

Q6. What is a MAC address? (CO4, K1)
Ans: A MAC address is a unique identifier assigned to every Network Interface Card (NIC). It consists of 48 bits usually represented in hexadecimal format. The first part identifies the manufacturer, while the latter part is unique to the device. MAC addresses are crucial in Ethernet communication.

Q7. Define ARP. (CO4, K1)
Ans: Address Resolution Protocol (ARP) maps IP addresses to their corresponding MAC addresses within a local network. When a device wants to communicate, it sends an ARP request asking who owns a specific IP. The target replies with its MAC address. This process enables correct packet delivery at the data link layer.

Q8. What is ARP cache poisoning? (CO4, K2)

Ans: ARP cache poisoning is an attack where false ARP responses are injected into a host’s ARP table. As a result, IP addresses map to incorrect MAC addresses controlled by the attacker. This allows attackers to intercept or redirect communication. It is often used to launch man-in-the-middle attacks.

Q9. How does ARP poisoning enable man-in-the-middle attacks? (CO4, K3)
Ans: ARP poisoning misdirects traffic by making devices believe the attacker’s machine is the legitimate destination. Once in the middle, attackers can intercept, modify, or drop packets. This enables eavesdropping on sensitive information such as passwords or financial data. The attack exploits ARP’s lack of authentication.

Q10. What is a virtual network interface? (CO4, K2)
Ans: A virtual network interface is a software-based NIC that exists without physical hardware. It is widely used in virtualization environments and containers to simulate network connections. Each virtual NIC has its own MAC and IP address. They provide isolation, scalability, and flexibility for network testing and deployments.

Q11. What is the purpose of the TTL field in an IP header? (CO4, K1)
Ans: The Time-To-Live (TTL) field limits the lifespan of a packet in the network. It is decremented at each hop, and the packet is discarded when it reaches zero. This prevents packets from circulating indefinitely in case of routing loops. Tools like traceroute use TTL to map network paths.

Q12. Define IP fragmentation. (CO4, K1)
Ans: IP fragmentation occurs when large packets are split into smaller fragments to fit the MTU of networks. Each fragment carries identification and offset values for reassembly at the destination. This allows data to traverse networks with varying MTU sizes. However, it also introduces risks of fragmentation-based attacks.

Q13. How can attackers misuse IP fragmentation? (CO4, K3)
Ans: Attackers send overlapping or inconsistent fragments to confuse the reassembly process. Firewalls or intrusion detection systems may fail to detect malicious payloads hidden in fragments. This technique can bypass security filters or cause buffer overflows. Such misuse can lead to denial-of-service or hidden malware injection.

Q14. What is an ICMP redirect attack? (CO4, K2)
Ans: ICMP redirect attacks involve sending fake ICMP messages to modify a host’s routing table. The victim is tricked into sending traffic through a malicious gateway. This allows the attacker to monitor or alter communications. The attack takes advantage of trust in ICMP messages.

Q15. Explain reverse path filtering in routing. (CO4, K2)
Ans: Reverse Path Filtering (RPF) is a security technique to prevent IP spoofing. It checks whether a packet’s source address is reachable via the interface it arrived on. If not, the packet is dropped as spoofed. RPF is commonly deployed on routers to mitigate DoS and spoofing attacks.

Q16. What is packet sniffing? (CO4, K1)
Ans: Packet sniffing is the practice of capturing network traffic for analysis. It is useful for diagnosing problems and monitoring security. Attackers, however, use sniffing to steal sensitive data such as passwords. Tools like Wireshark and tcpdump are widely used for both defense and offense.

Q17. Differentiate between raw sockets and pcap API in sniffing. (CO4, K2)
Ans: Raw sockets provide direct access to packet data at the operating system level. In contrast, the pcap API offers a higher-level, user-friendly interface with filtering features. While raw sockets are more flexible, pcap ensures better portability and ease of use. Both are essential for security research.

Q18. What is packet spoofing? (CO4, K1)
Ans: Packet spoofing is forging packet headers to impersonate another system. By modifying source IP or MAC addresses, attackers hide their identity. Spoofed packets can bypass filters and redirect communication. It is commonly used in DoS and session hijacking attacks.

Q19. How does sniffing help in spoofing? (CO4, K3)
Ans: Sniffing provides crucial details like session IDs and sequence numbers. Attackers use this information to craft spoofed packets resembling legitimate traffic. This allows injection of malicious data into ongoing sessions. Without sniffing, spoofing attempts often fail due to missing parameters.

Q20. Mention two tools for packet sniffing. (CO4, K1)
Ans: Wireshark and Scapy are two widely used tools. Wireshark provides a GUI-based interface for analyzing packets in detail. Scapy is a Python-based tool for crafting, sniffing, and replaying packets. Both are used for security testing, monitoring, and academic research.

Q21. What is a SYN flooding attack? (CO4, K2)
Ans: A SYN flood is a denial-of-service attack exploiting the TCP handshake. Attackers send many SYN requests without sending final ACKs. This exhausts the server’s backlog queue and denies service to legitimate users. It is one of the earliest and most effective DoS techniques.

Q22. What is the TCP three-way handshake? (CO4, K1)
Ans: The TCP handshake establishes a reliable connection using SYN, SYN-ACK, and ACK messages. The client initiates with SYN, the server replies with SYN-ACK, and the client responds with ACK. This sequence ensures both parties are ready for communication. It prevents packet loss during transmission.

Q23. Explain TCP reset attack. (CO4, K2)
Ans: In a TCP reset attack, the attacker sends forged RST packets to both ends of a session. This causes the connection to terminate immediately. The attacker only needs to guess or sniff the sequence number to succeed. It disrupts services like telnet, SSH, or streaming.

Q24. What is TCP session hijacking? (CO4, K3)

Ans: TCP session hijacking occurs when an attacker takes over an active TCP connection. By predicting sequence numbers or sniffing traffic, attackers inject malicious packets. They gain unauthorized access to the session, bypassing authentication. This can lead to data theft or further exploits like reverse shells.

Q25. State the significance of the Mitnick attack. (CO4, K2)
Ans: The Mitnick attack is a classic case of TCP spoofing and session hijacking. Kevin Mitnick exploited weak sequence number generation to impersonate trusted hosts. He silenced legitimate servers and injected spoofed packets to gain access. It highlighted flaws in early TCP/IP implementations.

Q26. What is DNS cache poisoning? (CO4, K2)
Ans: DNS cache poisoning inserts false entries into a resolver's cache. As a result, users are redirected to malicious websites despite entering correct domain names. Attackers exploit weak transaction ID generation to inject fake replies. This attack compromises user trust in the DNS system.

Q27. What is the Kaminsky attack? (CO4, K2)
Ans: The Kaminsky attack is a high-profile DNS poisoning method. It floods a resolver with queries for random subdomains while sending spoofed replies. Once a forged record is cached, attackers can control entire domains. This attack exposed the urgent need for DNSSEC deployment.

Q28. Define DNS rebinding attack. (CO4, K2)
Ans: DNS rebinding exploits the short TTL values of DNS records. Attackers use it to make browsers treat malicious and local hosts as the same origin. This allows bypassing of the Same-Origin Policy. It enables attacks on internal networks and IoT devices.

Q29. What is DNSSEC? (CO4, K2)
Ans: DNS Security Extensions (DNSSEC) provide authenticity and integrity for DNS responses. It uses public-key cryptography to digitally sign DNS records. Resolvers verify signatures before accepting records. This prevents cache poisoning and replay attacks.

Q30. How does a denial-of-service attack affect DNS servers? (CO4, K3)
Ans: In a DoS attack, attackers flood DNS servers with overwhelming queries or amplified traffic. Root, TLD, or authoritative servers may become unreachable. This disrupts name resolution for users globally. DNS DoS attacks can cripple websites and online services.

Part B Q

PART -B

Q1. Explain the life cycle of a packet with neat diagrams. Discuss the steps of packet construction, transmission, reception, and forwarding in detail. (CO4, K2)

Q2. Describe the structure of an Ethernet frame. Explain how ARP works and analyze how ARP cache poisoning leads to man-in-the-middle attacks. (CO4, K3)

Q3. With a neat diagram, explain the IP header fields. How are TTL and fragmentation fields exploited in attacks? (CO4, K2)

Q4. Discuss in detail the methods of packet sniffing using sockets, raw sockets, pcap API, and Scapy. Highlight their differences with examples. (CO4, K2)

Q5. Analyze SYN flooding attacks. Explain the TCP three-way handshake and describe how attackers exploit it with examples and countermeasures. (CO4, K4)

Q6. What is TCP session hijacking? Explain the steps of launching this attack, its impact on communication, and possible defenses. (CO4, K3)

Q7. Explain the concept of DNS cache poisoning with examples. Compare local cache poisoning and remote poisoning attacks such as the Kaminsky attack. (CO4, K4)

Q8. Describe DNS rebinding attacks. Explain their working, setup, and how they exploit Same-Origin Policy (SOP). Suggest effective countermeasures. (CO4, K4)

Q9. What is DNSSEC? Explain the working of DNSSEC with key generation, zone signing, and validation. How does it defend against DNS-based attacks? (CO4, K3)

Q10. Discuss denial-of-service (DoS) attacks on DNS servers. Differentiate between attacks on root/TLD servers and those on specific domain servers. Suggest prevention strategies. (CO4, K4)

Supportive Online Certification courses

SUPPORTIVE ONLINE COURSES

S No	Course provider	Course title	Link
1	Udemy	The Complete Cyber Security Course : Network Security	https://www.udemy.com/course/network-security-course/?couponCode=ST12MT90625AI
2	Udemy	Snort Intrusion Detection, Rule Writing, and PCAP Analysis	https://www.udemy.com/course/snort-intrusion-detection-rule-writing-and-pcap-analysis/?couponCode=ST12MT90625AI
3	NPTEL	Network Security	https://onlinecourses.nptel.ac.in/noc25_ee54/preview

Real life Applications in day to day life and to Industry

REAL TIME APPLICATIONS IN DAY TO DAY LIFE AND TO INDUSTRY

1. Day-to-Day Life – Online Banking (Protecting Users from DNS Attacks)

When a person accesses an online banking website (e.g., https://mybank.com), DNS resolves the domain name to the bank's IP address.

Without security, attackers could launch a DNS cache poisoning attack, redirecting the user to a fake website. The victim might unknowingly enter login credentials, losing sensitive data.

To protect against this, banks use DNSSEC to digitally sign DNS responses, ensuring that users only connect to the authentic site.

Additionally, packet sniffing protection (TLS/SSL encryption) ensures that even if someone captures packets on public Wi-Fi, the sensitive data like account numbers and passwords remain secure.

➔ This shows how network security ensures safe financial transactions in daily life.

2. Industry – Cloud Services and Data Centers (Preventing TCP & DoS Attacks)

A cloud service provider like AWS or Azure manages thousands of client applications running on servers.

Attackers may attempt a SYN flooding attack on these servers by sending half-open TCP requests. This overwhelms the server and prevents legitimate clients from accessing services.

To defend, cloud providers implement SYN cookies, firewalls, and intrusion detection systems to handle abnormal traffic.

They also use reverse path filtering and anti-spoofing techniques at routers to block malicious IP packets.

➔ This ensures high availability and reliability of industry-scale services, preventing downtime and financial loss.

Content beyond Syllabus

Contents beyond the Syllabus

2. Zero Trust Network Security (ZTNA): https://www.cloudflare.com/learning/security/glossary/what-is-zero-trust/

Assessment Schedule

FIAT: 14.08.2025

SIAT: 23.09.2025

Model Exam: 28.10.2025

Prescribed Text books & Reference books

PRESCRIBED TEXT BOOKS AND REFERENCE BOOKS

TEXT BOOKS

1. Rafeeq Rehman : “Intrusion Detection with SNORT, Apache, MySQL, PHP and ACID,” 1st Edition, Prentice Hall , 2003.

2. Internet Security: A Hands-on Approach, by Wenliang Du, Third Edition, 2019

REFERENCE BOOKS

1. Christopher Kruegel, Fredrik Valeur, Giovanni Vigna: “Intrusion Detection and Correlation Challenges and Solutions”, 1st Edition, Springer, 2005.
2. Carl Endorf, Eugene Schultz and Jim Mellander “Intrusion Detection & Prevention”, 1st Edition, Tata McGraw-Hill, 2004.
3. Stephen Northcutt, Judy Novak : “Network Intrusion Detection”, 3rd Edition, New Riders Publishing, 2002.
4. T. Fahringer, R. Prodan, “A Text book on Grid Application Development and Computing Environment”. 6th Edition, Khanna Publishers, 2012.

Mini ProjectSuggestions

MINI PROJECT SUGGESTIONS

1. Packet Sniffer using Python – Build a program to capture and analyze packets (like Wireshark). (CO4, K3 – Apply)

2. ARP Cache Poisoning Demo – Simulate ARP spoofing in a lab and test prevention with arpwatch/static ARP.(CO4, K4 – Analyze)

3. SYN Flooding Attack Simulation – Launch a SYN flooding attack on a test server and defend with SYN cookies/firewall. (CO4, K4 – Analyze)

4. DNS Spoofing & DNSSEC – Perform a DNS cache poisoning attack and secure it using DNSSEC validation. (CO4, K5 – Evaluate)

5. Secure Chat Application – Create a chat app with encryption (AES/RSA) to prevent sniffing/spoofing. (CO4, K5 – Evaluate & Create)

Thank you

Disclaimer:

This document is confidential and intended solely for the educational purpose of RMK Group of Educational Institutions. If you have received this document through email in error, please notify the system manager. This document contains proprietary information and is intended only to the respective group / learning community as intended. If you are not the addressee you should not disseminate, distribute or copy through e-mail. Please notify the sender immediately by e-mail if you have received this document by mistake and delete this document from your system. If you are not the intended recipient you are notified that disclosing, copying, distributing or taking any action in reliance on the contents of this information is strictly prohibited.