Week 12 revision notes

IFB240 Week 12 - Network Security

Overview

This week focuses on **network security protocols**, such as **IPSec**, **TLS**, **SSH**, and **HTTP** security mechanisms like **Basic** and **Digest Authentication**. We discuss the functioning of these protocols, the modes of operation, the security services they provide, and the real-world applications and limitations involved.

1. HTTP and Network Communication Protocols

What is HTTP?

- **Hypertext Transfer Protocol (HTTP)**:
  - An **application layer** protocol used for transferring data between clients (web browsers) and servers.
  - Follows a **request/response** model:
    1. **Client** requests access to a resource (e.g., webpage).
    2. **Server** responds with the resource or an error message.
  - **Port**: HTTP uses **port 80**.

HTTP Authentication

- **Basic Authentication**:
  - The **username** and **password** are encoded using **Base64** and sent in the request header.
  - **Security Concern**: This mechanism is essentially **plaintext** transmission, and an attacker can easily intercept and decode credentials.
- **Digest Authentication**:
  - More secure than Basic Authentication, as it involves a **challenge-response** mechanism.
  - Uses a **nonce** (random value), and the client sends a hash of the **username, password, and nonce**.
  - **Security Concern**: Vulnerable to dictionary attacks if users choose weak passwords.

Practical Example - How to Use HTTP Authentication

1. **Basic Authentication**:

- **Client sends request** for a resource.
- **Server responds with 401 Unauthorized** and asks for credentials.
- **Client resends request** with **Base64 encoded credentials** (username:password).
- **Server verifies credentials** and grants access if correct.

2. **Digest Authentication**:
   - **Client requests resource**.
   - **Server sends 401 Unauthorized** with a **challenge**.
   - **Client computes response digest** based on **username, password, nonce**, and resends the request with this digest.
   - **Server compares digests** and grants access if they match.

2. SSH - Secure Shell

What is SSH?

- **Secure Shell (SSH)** is a **cryptographic protocol** for securing communication over a network.
- Used for **remote login**, **file transfer**, and **command execution**.
- SSH uses **port 22** and provides encrypted communication.

SSH Protocol Components

1. **SSH Transport Layer Protocol**:
   - Provides **server authentication**, **confidentiality**, and **data integrity**.
   - **Key Exchange**: Diffie-Hellman (DH) is commonly used for generating shared keys.
2. **SSH User Authentication Protocol**:
   - Authenticates the **user** to the **server**.
   - **Authentication Methods**:
     - **Password-Based**: Requires the server to store and validate the user's password.
     - **Public Key Authentication**: The client generates a **public-private key pair** and sends the **public key** to the server, which uses it to verify the signature.
3. **SSH Connection Protocol**:
   - Allows **multiple logical communications** over a single SSH connection.
   - Supports protocols like **SFTP** for secure file transfers.

How to Set Up SSH Authentication

1. **Password-Based Authentication**:
   - Server stores and verifies user passwords.
   - Less secure, as the password could be exposed or compromised.

2. **Public Key Authentication**:
    - **Client**: Generates an RSA or ECC **key pair**.
    - **Server**: Stores the **public key**.
    - **Private Key Protection**: The private key should be kept secure, ideally encrypted with a passphrase.

3. TLS - Transport Layer Security

What is TLS?

- **TLS** is a cryptographic protocol used for secure end-to-end communications, often used to secure HTTP (i.e., **HTTPS**).
- Operates between the **application** and **transport** layers.

TLS Handshake and Record Protocols

1. **TLS Handshake Protocol**:
    - **Negotiates cryptographic algorithms** to use.
    - **Authenticates** the server to the client using digital certificates.
    - **Generates session keys** for encrypting further communication.
2. **TLS Record Protocol**:
    - Provides **confidentiality** by encrypting data before transmission.
    - Ensures **integrity** using **MAC** to detect tampering.

Example - How TLS Works with HTTPS

1. **Client sends "Hello"** to initiate the handshake.
2. **Server sends certificate** and **public key**.
3. **Client generates pre-master secret**, encrypts it with the server's public key, and sends it to the server.
4. **Both entities derive session keys** to be used for symmetric encryption.
5. The **HTTP request and response** are encrypted using this session key.

HTTPS and Ports

- **HTTP**: Uses **port 80**.
- **HTTPS** (HTTP over TLS): Uses **port 443**.
- The **TLS handshake** ensures the connection is secure before HTTP communication begins.

4. IPSec - Internet Protocol Security

What is IPSec?

- **IPSec** is a framework of protocols for securing IP communications by **authenticating** and **encrypting** each IP packet.
- Operates at the **network layer**.

IPSec Protocols

1. **Internet Key Exchange (IKE)**:
   - Used to negotiate and establish **Security Associations (SAs)**.
   - Agrees on cryptographic algorithms, keys, and sequence numbers.
2. **Encapsulating Security Payload (ESP)**:
   - Provides **confidentiality**, **integrity**, and **authentication** for IP packets.
   - Encrypts the packet's payload.
3. **Authentication Header (AH)**:
   - Provides **authentication** and **integrity** for the packet, including some header fields.
   - No confidentiality provided.

Modes of Operation

1. **Transport Mode**:
   - **Secures only the payload** of the packet.
   - Suitable for **host-to-host** communication, commonly used for **end-to-end encryption**.
2. **Tunnel Mode**:
   - **Encapsulates the entire packet**, including the original header, as the payload of a new packet.
   - Used for **gateway-to-gateway** or **host-to-gateway** architectures, often for **VPNs**.

IPSec Architectures

1. **Gateway-to-Gateway**:
   - Secures communication between **two secured networks** via gateways.
   - Useful for linking **branch offices** securely over the internet.
2. **Host-to-Gateway**:
   - Secures communication between a **single host** and a secure network via a gateway.
   - Suitable for employees accessing corporate resources remotely.
3. **Host-to-Host**:
   - Provides **end-to-end security** between **two individual hosts**.
   - Requires all systems to have IPSec configured, and it is resource-intensive to manage.

How to Use IPSec

1. **Configure Security Associations** using **IKE**.
2. **Choose the appropriate mode** (Transport or Tunnel) based on the architecture.
3. **Establish the VPN** connection:
    - For a **gateway-to-gateway** VPN, configure each gateway with the other's IP address.
    - For **host-to-gateway**, set up the host's VPN client to connect to the gateway's IP.

5. Summary and Key Takeaways

- **HTTP Authentication**: Basic and Digest methods for securing access to web resources, but vulnerable to various attacks.
- **SSH**: A secure protocol for remote login, using encryption to protect the confidentiality and integrity of the connection.
- **TLS**: Provides a secure communication channel for protocols like **HTTPS**, ensuring **confidentiality**, **integrity**, and **authentication**.
- **IPSec**: A set of protocols that protect IP communications, either in **Transport** or **Tunnel Mode**, providing security for networks and VPNs.

Practical Tips

- Avoid **Basic Authentication** in HTTP for sensitive resources; use **TLS** to secure the communication channel.
- Use **public key authentication** for SSH to enhance security compared to password-based methods.
- For securing network communications, implement **IPSec** in tunnel mode for secure VPN connections.

These notes provide a comprehensive understanding of the essential protocols used to secure network communications, detailing both theory and practical application.