|  |
| --- |
| **1: IP sec Architecture** |

**Definition:**  
IPSec is a framework of protocols designed to secure data transmitted over IP networks by providing encryption, authentication, and integrity.

**Key Components:**

**Authentication Header (AH):** Verifies the integrity and authenticity of the data.

**Encapsulating Security Payload (ESP):** Encrypts data for confidentiality and can also provide authentication.

**Standards:**

Defined by several RFCs, including:

RFC 2401: Overview of IPSec architecture.

RFC 2402: Packet authentication.

RFC 2406: Packet encryption.

RFC 2408: Key management.

**Implementation:**  
IPSec works as extension headers in IP packets, added after the main IP header. It is mandatory in IPv6 and optional in IPv4.

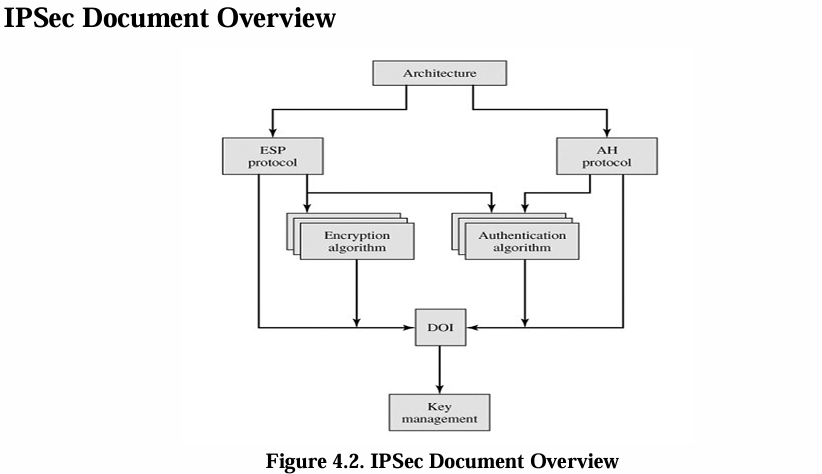
**Services:**  
IPSec ensures:

Confidentiality: Protects data from unauthorized access.

Integrity: Ensures data is not altered during transmission.

Authentication: Verifies the sender’s identity.

Key Management: Safely generates and exchanges cryptographic keys.

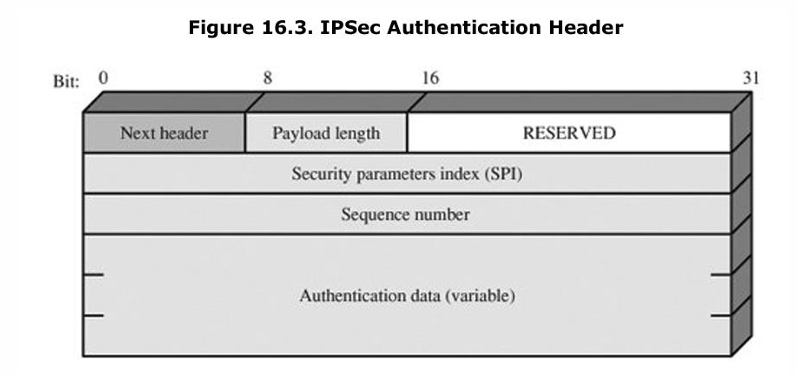


specific aspects of the protocol. Here’s a detailed breakdown of the groups:

1. **Architecture**:
   * Focuses on the general concepts and framework of IPSec.
   * Covers the **security requirements**, definitions, and mechanisms that define IPSec technology.
   * Provides the foundational understanding necessary for implementing IPSec.
2. **Encapsulating Security Payload (ESP)**:
   * Details the **packet format** used by ESP.
   * Addresses issues related to **packet encryption** and the optional feature of authentication provided by ESP.
   * Ensures data confidentiality and can optionally verify integrity and authenticity.
3. **Authentication Header (AH)**:
   * Focuses on the **packet format** and issues related to the use of AH.
   * Describes how AH ensures **packet authentication** and data integrity.
   * Protects against tampering and replay attacks without providing confidentiality.
4. **Encryption Algorithm**:
   * A set of documents describing how various **encryption algorithms** are implemented for ESP.
   * Includes methods like **DES, Triple DES, AES**, and others used to ensure data confidentiality.
   * Specifies how these algorithms are applied for different scenarios.
5. **Authentication Algorithm**:
   * A set of documents focusing on **authentication algorithms** used for AH and ESP’s authentication feature.
   * Includes algorithms like **HMAC-MD5** and **HMAC-SHA-1** to verify data integrity and authenticity.
   * Defines how these methods are incorporated into IPSec protocols.
6. **Key Management**:
   * Documents describing schemes for managing **cryptographic keys**.
   * Includes protocols like **Oakley** and **ISAKMP**, which are essential for secure key exchange and management.
   * Ensures keys are exchanged securely and remain fresh during communication.
7. **Domain of Interpretation (DOI)**:
   * Defines values that link other IPSec documents together.
   * Contains identifiers for **approved encryption and authentication algorithms**.
   * Specifies operational parameters like **key lifetimes** to ensure consistency across implementations.

These categories collectively form the basis of IPSec’s robus

|  |
| --- |
| 2: IP sec header |



The **Authentication Header (AH)** fields are crucial for ensuring data integrity, authenticity, and protection against replay attacks. Here’s a detailed explanation of its components:

1. **Next Header (8 bits)**:
   * Identifies the type of header that immediately follows the AH.
   * Commonly indicates whether the next header is a TCP, UDP, or another IP header.
2. **Payload Length (8 bits)**:
   * Specifies the length of the AH in **32-bit words**, minus 2.
   * For example, if the default length of the **Authentication Data** field is **96 bits** (3 × 32-bit words), the entire header length becomes six 32-bit words (including the fixed header).
   * In this case, the **Payload Length** value will be **4**.
3. **Reserved (16 bits)**:
   * Reserved for future use and must be set to zero.
4. **Security Parameters Index (SPI) (32 bits)**:
   * Identifies a **Security Association (SA)**.
   * An SA defines the parameters (e.g., encryption and authentication methods) used to secure a communication session between two parties.
5. **Sequence Number (32 bits)**:
   * A monotonically increasing counter to prevent **replay attacks**.
   * Each AH packet is assigned a unique sequence number, ensuring that old or duplicate packets can be detected and discarded.
6. **Authentication Data (variable)**:
   * A variable-length field containing the **Integrity Check Value (ICV)**, also known as the **Message Authentication Code (MAC)**.
   * The length must be a multiple of **32-bit words**.
   * This value is computed using cryptographic hash functions (e.g., HMAC-SHA-1, HMAC-MD5) and ensures that the packet’s integrity and authenticity remain intact.

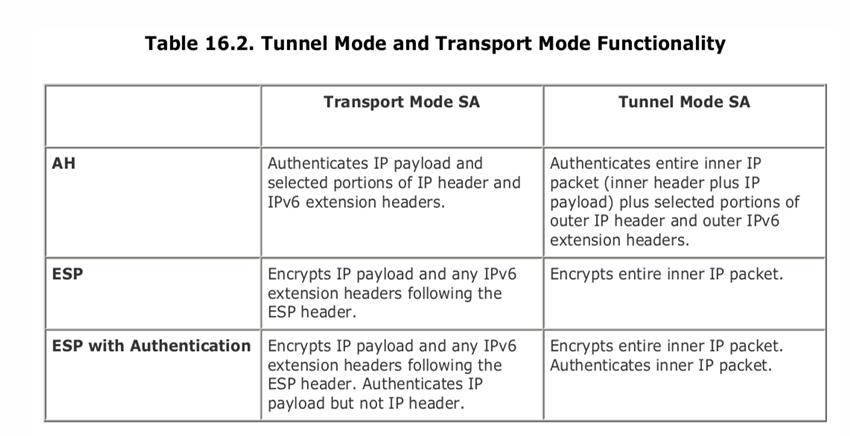
|  |
| --- |
| 3: Transport mode and Tunnel Mode |

**Transport Mode**

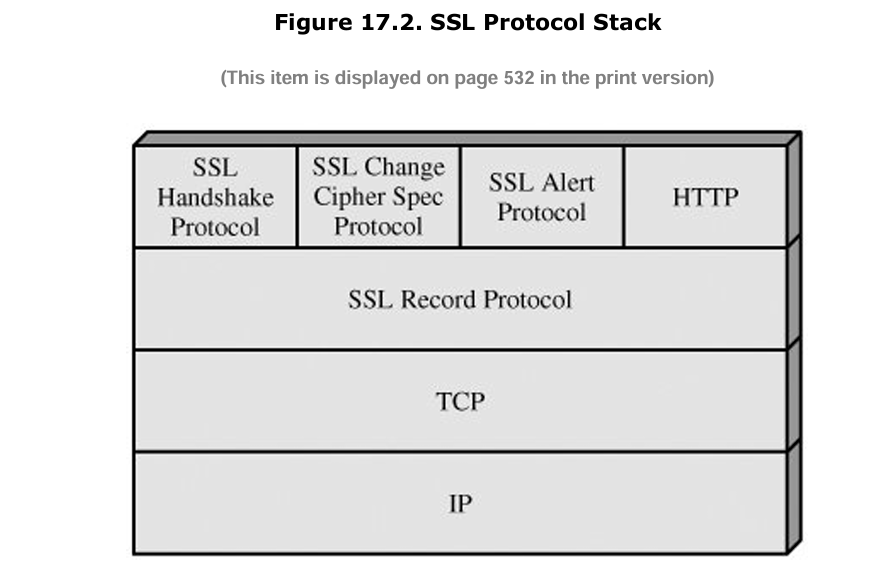
* Transport Mode provides protection primarily for upper-layer protocols, extending to the payload of an IP packet.
* Examples of payloads include TCP or UDP segments or ICMP packets, which operate directly above IP in a host protocol stack.
* Typically used for end-to-end communication between two hosts, such as a client and a server or two workstations.
* For IPv4, the payload is the data that follows the IP header.
* For IPv6, the payload includes data following the IP header and any IPv6 extension headers, except possibly the destination options header.
* ESP in transport mode encrypts and optionally authenticates the IP payload but does not protect the IP header.
* AH in transport mode authenticates the IP payload and selected portions of the IP header.

**Tunnel Mode**

* Tunnel Mode protects the entire IP packet by encapsulating it with a new outer IP header.
* Intermediate routers cannot access the inner IP header, enhancing security.
* Used when one or both ends of an SA are security gateways like firewalls or routers.
* Hosts behind firewalls can communicate securely without implementing IPSec; the firewall handles tunnelling.
* Example: A firewall encapsulates a packet from Host A to Host B with an outer header; Host B's firewall removes it and delivers the inner packet.
* ESP encrypts and optionally authenticates the entire inner packet.
* AH authenticates the inner packet and parts of the outer header.



|  |
| --- |
| 6: SSL Protocol Stack |



This **SSL Protocol Stack** diagram illustrates the structure of the SSL (Secure Sockets Layer) protocol, showing how it works in layers to provide secure communication over a network. Let’s break it down:

**1. Top Layer: Application Layer Protocols**

At the top of the SSL stack are the **application protocols** that rely on SSL for secure communication.

* **HTTP**: Hypertext Transfer Protocol, the protocol used for web browsing. When combined with SSL, it becomes HTTPS, which provides encryption and secure transmission of web data.
* **SSL Handshake Protocol**: This protocol is responsible for establishing and negotiating a secure session between a client and a server. It includes:
  + Authentication (via certificates)
  + Negotiation of encryption algorithms
  + Key exchange.
* **SSL Change Cipher Spec Protocol**: A simple protocol that signals that the agreed cryptographic parameters (such as encryption keys) are ready to be used.
* **SSL Alert Protocol**: Used for reporting errors and warnings during an SSL session, such as bad certificates, connection termination, or handshake failures.

**2. SSL Record Protocol**

The **SSL Record Protocol** is the core of SSL and sits below the higher-level protocols. It is responsible for:

* **Fragmentation**: Breaking down data into manageable blocks.
* **Compression**: Optionally compressing the data (before encryption).
* **Encryption**: Encrypting the data for confidentiality.
* **Message Authentication**: Adding a MAC (Message Authentication Code) to ensure integrity.

The SSL Record Protocol ensures that all data transmitted between the client and server remains confidential and tamper-proof.

**3. Transport Layer: TCP**

SSL runs on top of the **TCP (Transmission Control Protocol)**, which provides a reliable, connection-oriented transport service:

* It ensures data packets are delivered in order and without loss.
* SSL uses TCP for its reliability because secure connections need dependable delivery of messages.

**4. Network Layer: IP**

The lowest layer shown in the diagram is **IP (Internet Protocol)**, which handles the actual delivery of packets over the network.

* IP is responsible for addressing and routing packets between sender and receiver.

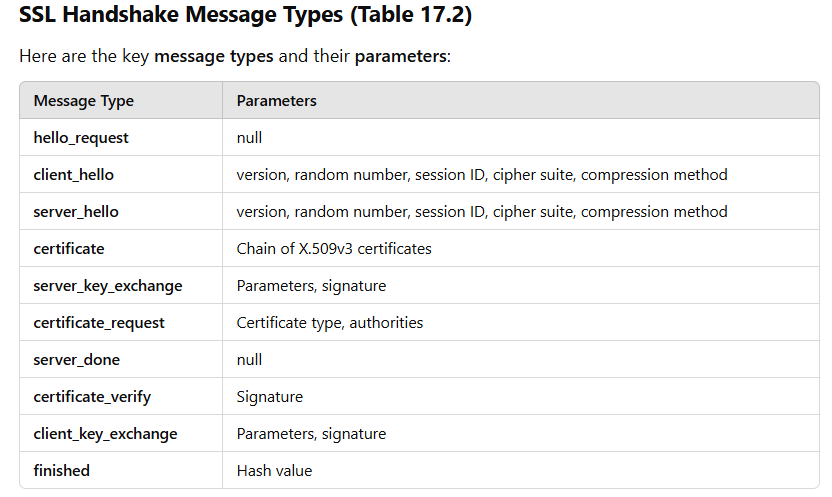
|  |
| --- |
| 7: Handshake Protocol Action |

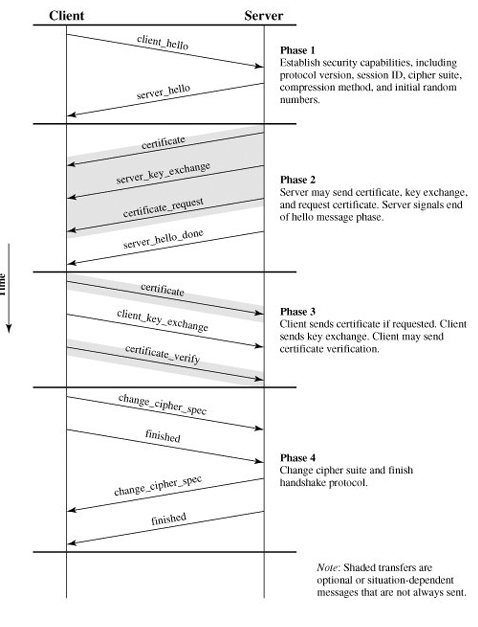
The **SSL Handshake Protocol** is the most complex part of SSL because it enables secure communication between the **client** and **server** before any application data is transmitted. It is designed to authenticate the client and server, agree on cryptographic algorithms, and establish encryption keys. Let’s break this down further:

**Handshake Message Format**

Each message exchanged during the SSL handshake has the following structure:

1. **Type (1 byte)**: Specifies the type of message (one of 10 defined types listed in **Table 17.2**).
2. **Length (3 bytes)**: Indicates the size of the message in bytes.
3. **Content (variable size)**: Contains parameters specific to the message type.





**Phase 1: Client and Server Hello**

This phase initiates the SSL handshake and sets up the basic parameters for the session.

* **Client Hello**: The client sends a message that includes:
  + Protocol version (e.g., SSL 3.0, TLS 1.2, etc.)
  + Session ID
  + Cipher suites supported (encryption algorithms)
  + Compression methods
  + Random number (used in key generation).
* **Server Hello**: The server responds with:
  + Protocol version selected
  + Session ID
  + Cipher suite chosen from the client's list
  + Compression method
  + A random number.

This phase establishes the security capabilities to be used during the session.

**Phase 2: Server Certificate and Key Exchange**

The server shares its certificate and optionally requests the client's certificate for mutual authentication.

* **Certificate**: The server sends its digital certificate, which contains the server's public key. The client uses this to verify the server's identity.
* **Server Key Exchange**: If necessary (e.g., when using **Diffie-Hellman** or certain cipher suites), the server sends additional key exchange data.
* **Certificate Request**: The server optionally requests the client's certificate (for client authentication).
* **Server Hello Done**: Indicates the server has completed its part of the handshake.

**Phase 3: Client Certificate and Key Exchange**

The client completes the handshake by providing key exchange and optionally verifying itself.

* **Certificate**: The client sends its certificate (if requested in Phase 2).
* **Client Key Exchange**: The client sends key exchange information. This is critical for establishing the shared secret (e.g., pre-master secret), which will be used to derive the encryption keys.
* **Certificate Verify**: If the client sent a certificate, it may provide a verification message to prove ownership of the private key associated with the certificate.

**Phase 4: Change Cipher Spec and Finish**

The session parameters are finalized, and the handshake concludes.

* **Change Cipher Spec**: Both the client and server signal that subsequent messages will be encrypted using the agreed cipher suite and shared keys.
* **Finished**: Each side sends a “Finished” message, which is encrypted using the new session keys. This ensures that the handshake was successful and no tampering occurred.

|  |
| --- |
| 8: SET: features, benefits, requirements and Secure Electronic Commerce components |

**SET Overview**

A good way to begin the discussion of SET is by looking at the **business requirements**, **key features**, and the **participants** in SET transactions.

**Business Requirements**

Book 1 of the SET specification lists the following **business requirements** for secure payment processing with credit cards over the Internet and other networks:

1. **Provide Confidentiality of Payment and Ordering Information**
   * It is necessary to ensure that payment and ordering information is safe and accessible only to the intended recipient.
   * Confidentiality also reduces the risk of fraud by either party to the transaction or by malicious third parties.
   * **SET** uses encryption to provide confidentiality.
2. **Ensure the Integrity of All Transmitted Data**
   * Ensure that no changes in content occur during the transmission of SET messages.
   * **Digital signatures** are used to provide integrity.
3. **Provide Authentication That a Cardholder is a Legitimate User of a Credit Card Account**
   * A mechanism that links a cardholder to a specific account number reduces the incidence of fraud and the overall cost of payment processing.
   * **Digital signatures** and **certificates** are used to verify that a cardholder is a legitimate user of a valid account.
4. **Provide Authentication That a Merchant Can Accept Credit Card Transactions**
   * This ensures cardholders can identify merchants with whom they can conduct secure transactions.
   * Again, **digital signatures** and **certificates** are used to verify the merchant’s legitimacy.
5. **Ensure the Use of the Best Security Practices and System Design Techniques**
   * **SET** is a well-tested specification based on highly secure cryptographic algorithms and protocols to protect all legitimate parties in an electronic commerce transaction.
6. **Create a Protocol That Does Not Depend on Transport Security Mechanisms**
   * **SET** can securely operate over a "raw" TCP/IP stack without depending on other transport security mechanisms.
   * However, **SET** does not interfere with the use of other security mechanisms, such as **IPSec** and **SSL/TLS**.
7. **Facilitate and Encourage Interoperability Among Software and Network Providers**
   * The **SET** protocols and formats are independent of hardware platform, operating system, and Web software.

**Key Features of SET**

To meet the requirements outlined earlier, **SET** incorporates the following features:

1. **Confidentiality of Information**
   * Cardholder account and payment information is secured as it travels across the network.
   * An important feature of **SET** is that it prevents the merchant from learning the cardholder's credit card number; this is only provided to the issuing bank.
   * **Conventional encryption by DES** is used to provide confidentiality.
2. **Integrity of Data**
   * Payment information sent from cardholders to merchants includes order information, personal data, and payment instructions.
   * **SET** guarantees that these message contents are not altered in transit.
   * **RSA digital signatures**, using **SHA-1** hash codes, provide message integrity. Certain messages are also protected by **HMAC** using **SHA-1**.
3. **Cardholder Account Authentication**
   * **SET** enables merchants to verify that a cardholder is a legitimate user of a valid card account number.
   * **SET** uses **X.509v3 digital certificates** with **RSA signatures** for this purpose.
4. **Merchant Authentication**
   * **SET** enables cardholders to verify that a merchant has a relationship with a financial institution allowing it to accept payment cards.
   * **SET** uses **X.509v3 digital certificates** with **RSA signatures** for this purpose.

Note: Unlike **IPSec** and **SSL/TLS**, **SET** provides only one choice for each cryptographic algorithm. This is because **SET** is a single application with a single set of requirements, whereas **IPSec** and **SSL/TLS** are intended to support a range of applications.

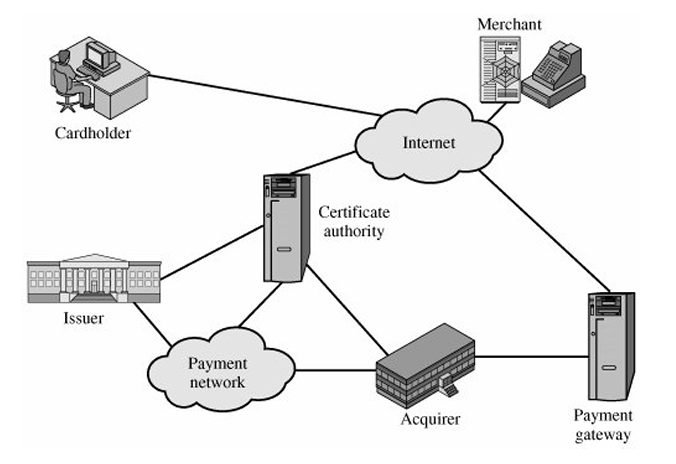
**SET Participants**

The following are the participants in the **SET** system (as depicted in Figure 17.8):

1. **Cardholder**
   * In the electronic environment, consumers and corporate purchasers interact with merchants from personal computers over the Internet.
   * A **cardholder** is an authorized holder of a payment card (e.g., MasterCard, Visa) that has been issued by an issuer.
2. **Merchant**
   * A **merchant** is a person or organization that has goods or services to sell to the cardholder. Typically, these goods and services are offered via a website or by electronic mail.
   * A merchant that accepts payment cards must have a relationship with an acquirer.
3. **Issuer**
   * The **issuer** is a financial institution, such as a bank, that provides the cardholder with the payment card.
   * Typically, accounts are applied for and opened by mail or in person. Ultimately, the issuer is responsible for the payment of the debt of the cardholder.
4. **Acquirer**
   * The **acquirer** is a financial institution that establishes an account with a merchant and processes payment card authorizations and payments.
   * Merchants will usually accept more than one credit card brand but do not want to deal with multiple bankcard associations or with multiple individual issuers.
   * The acquirer provides authorization to the merchant that a given card account is active and that the proposed purchase does not exceed the credit limit. The acquirer also provides electronic transfer of payments to the merchant's account. Subsequently, the acquirer is reimbursed by the issuer over some sort of payment network for electronic funds transfer.
5. **Payment Gateway**
   * The **payment gateway** is a function operated by the acquirer or a designated third party that processes merchant payment messages.
   * The payment gateway interfaces between **SET** and the existing bankcard payment networks for authorization and payment functions. The merchant exchanges **SET** messages with the payment gateway over the Internet, while the payment gateway has some direct or network connection to the acquirer's financial processing system.
6. **Certification Authority (CA)**
   * The **CA** is an entity trusted to issue **X.509v3 public-key certificates** for cardholders, merchants, and payment gateways.
   * The success of **SET** depends on the existence of a CA infrastructure available for this purpose. A hierarchy of CAs is used, so participants need not be directly certified by a root authority.

**Secure Electronic Commerce Components**

The sequence of events required for a transaction in **SET** is as follows:



1. **The customer opens an account**
   * The customer obtains a credit card account, such as MasterCard or Visa, with a bank that supports electronic payment and **SET**.
2. **The customer receives a certificate**
   * After suitable identity verification, the customer receives an **X.509v3 digital certificate** signed by the bank.
   * The certificate verifies the customer’s RSA public key and its expiration date. It also establishes a relationship, guaranteed by the bank, between the customer’s key pair and their credit card.
3. **Merchants have their own certificates**
   * A merchant who accepts a certain brand of card must possess two certificates for two public keys owned by the merchant: one for signing messages and one for key exchange.
   * The merchant also needs a copy of the **payment gateway's public-key certificate**.
4. **The customer places an order**
   * The process may involve the customer browsing the merchant’s website to select items and determine the price.
   * The customer sends a list of items to be purchased to the merchant, who returns an order form containing the list of items, their prices, the total price, and an order number.
5. **The merchant is verified**
   * In addition to the order form, the merchant sends a copy of its certificate to the customer, so they can verify that they are dealing with a valid store.
6. **The order and payment are sent**
   * The customer sends both order and payment information to the merchant, along with the customer’s certificate.
   * The order confirms the purchase of the items in the order form, and the payment contains credit card details.
   * The payment information is encrypted so that it cannot be read by the merchant. The customer’s certificate enables the merchant to verify the customer.
7. **The merchant requests payment authorization**
   * The merchant sends the payment information to the **payment gateway**, requesting authorization that the customer’s available credit is sufficient for this purchase.
8. **The merchant confirms the order**
   * The merchant sends confirmation of the order to the customer.
9. **The merchant provides the goods or service**
   * The merchant ships the goods or provides the service to the customer.
10. **The merchant requests payment**

* The request for payment is sent to the **payment gateway**, which handles all of the payment processing.

|  |
| --- |
| 9: Intrusion :Intrusion Detection |

**Intruders**

* One of the two most publicized threats to security is the intruder (the other is viruses), generally referred to as a hacker or cracker. In an important early study of intrusion, Anderson [ANDE80] identified three classes of intruders:
  + **Masquerader**: An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account.
  + **Misfeasor**: A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges.
  + **Clandestine user**: An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection.
* The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider.
* In fact, the problem has not been brought under control. To cite one example, a group at Bell Labs [BELL92, BELL93] has reported persistent and frequent attacks on its computer complex via the Internet over an extended period and from a variety of sources. At the time of these reports, the Bell group was experiencing the following:
  + Attempts to copy the password file (discussed later) at a rate exceeding once every other day.
  + Suspicious remote procedure call (RPC) requests at a rate exceeding once per week.
  + Attempts to connect to non-existent "bait" machines at least every two weeks.
* Benign intruders might be tolerable, although they do consume resources and may slow performance for legitimate users. However, there is no way in advance to know whether an intruder will be benign or malign. Consequently, even for systems with no particularly sensitive resources, there is a motivation to control this problem.

**Intrusion Techniques**

The primary goal of an intruder is to gain access to a system or expand their privileges within it. Typically, this requires acquiring information that should be protected, such as a user password. Once an intruder knows a password, they can log in to the system and gain the privileges of the legitimate user.

Systems often maintain a password file to link users to their credentials. If this file isn't protected, it becomes an easy target. There are two main ways to protect password files:

1. **One-way function**: The password is processed through a one-way transformation (irreversible), so even if an intruder gains access to the password file, they cannot directly recover the password.
2. **Access control**: Access to the password file is restricted to only a few authorized accounts.

If these countermeasures are in place, it still requires significant effort for an intruder to obtain passwords. Based on a study [ALVA90], the following techniques are commonly used to guess or retrieve passwords:

1. **Default passwords**: Many systems come with default passwords, which administrators may forget to change.
2. **Short passwords**: Intruders often attempt short passwords, ranging from one to three characters.
3. **Dictionary attack**: Trying common words from the system's dictionary or a list of known likely passwords.
4. **Social engineering**: Gathering personal information about users (e.g., names of family members, office details, etc.) to guess their passwords.
5. **Personal identifiers**: Using information like phone numbers, Social Security numbers, or room numbers.
6. **License plate numbers**: Guessing passwords based on common patterns like state license plate numbers.
7. **Trojan horses**: Programs that trick users into running malicious code, such as a game that secretly copies the password file.
8. **Line tapping**: Intercepting communication between a user and the host system, often requiring physical access but countered by encryption.

|  |
| --- |
| 10: Viruses: nature of worms, worms, state of work technology |

**Viruses**

A **computer virus** is a self-replicating program that attaches itself to legitimate files or software and spreads to other systems when executed. Viruses require human interaction (like opening an infected file or program) to propagate.

**Key Characteristics:**

1. **Self-replication**: They replicate by attaching to other files.
2. **Execution**: They need a host program to execute.
3. **Infection**: They infect files, programs, or boot sectors.
4. **Payload**: Some viruses are designed to cause harm, such as deleting files, corrupting data, or creating backdoors.

**Nature of Worms**

A **worm** is a standalone malware program that spreads itself across networks **without requiring a host program or human intervention**. Unlike viruses, worms exploit network vulnerabilities to propagate automatically.

**Key Characteristics:**

1. **Autonomous Propagation**: Worms don’t need user action to spread.
2. **Network Exploitation**: They leverage vulnerabilities in network protocols and services.
3. **Resource Consumption**: Worms often consume bandwidth, CPU, or memory, causing system slowdowns or crashes.
4. **Payload**: Some worms install additional malware, steal information, or disrupt services.

**Comparison: Viruses vs. Worms**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Viruses** | **Worms** |
| **Dependency** | Needs a host file/program to attach to. | Standalone program; no host needed. |
| **Propagation** | Requires user interaction to spread. | Spreads automatically over networks. |
| **Target** | Files, boot sectors, or programs. | Network systems or devices. |
| **Speed** | Slower due to dependency on users. | Faster, spreads autonomously. |

**State of Worm Technology**

1. **Exploitation of Vulnerabilities**: Modern worms exploit software vulnerabilities in **operating systems**, network protocols (e.g., TCP/IP), or services (e.g., SMB, FTP).
2. **Polymorphic Worms**:
   * These worms change their code or behavior with each infection to evade detection by antivirus software.
   * **Polymorphism** makes it harder to create virus definitions for detection.
3. **Network Propagation**: Worms use advanced propagation techniques:
   * **Email Worms**: Spread through malicious email attachments or links.
   * **Network Worms**: Exploit weak network defenses (e.g., SQL Slammer).
   * **P2P Worms**: Spread via peer-to-peer file-sharing networks.
4. **Advanced Payloads**:
   * Modern worms often deliver additional malware, such as **ransomware** or **spyware**.
   * They can steal data, monitor activities, or disrupt critical services.
5. **Zero-Day Exploits**: Worms increasingly use **zero-day vulnerabilities** (previously unknown software bugs) to infiltrate systems before patches are available.
6. **State-Sponsored Worms**:
   * Worms like **Stuxnet** demonstrate how advanced malware can be used in **cyber warfare** to sabotage critical infrastructure (e.g., nuclear facilities).