HASH FUNCTION

A **hash function** H accepts a variable-length block of data *M* as input and produces a fixed-size hash value *h* = H(*M*). A "good" hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random. In general terms, the principal object of a hash function is data integrity. A change to any bit or bits in *M* results, with high probability, in a change to the hash code. The kind of hash function needed for security applications is referred to as a **cryptographic hash function**.

Applications of Cryptographic Hash Functions

Perhaps the most versatile cryptographic algorithm is the cryptographic hash function. It is used in a wide variety of security applications and Internet protocols.

➢ Message Authentication

Message authentication is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay). When a hash function is used to provide message authentication, the hash function value is often referred to as a **message digest**.

The essence of the use of a hash function for message authentication is as follows. The sender computes a hash value as a function of the bits in the message and transmits both the hash value and the message. The receiver performs the same hash calculation on the message bits and compares this value with the incoming hash value. If there is a mismatch, the receiver knows that the message (or possibly the hash value) has been altered.

➢ Digital Signatures

Another important application, which is similar to the message authentication application, is the **digital signature**. The operation of the digital signature is similar to that of the MAC. In the case of the digital signature, the hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature.

➢ Other Applications

- Hash functions are commonly used to create a **one-way password file**. A hash of a password is stored by an operating system rather than the password itself. Thus, the actual password is not retrievable by a hacker who gains access to the password file. In simple terms, when a user enters a password, the hash of that password is compared to the stored hash value for verification.

- Hash functions can be used for **intrusion detection** and **virus detection**. Store H(F) for each file on a system and secure the hash values (e.g., on a CD-R that is kept secure). One can later determine if a file has been modified by recomputing H(F).

- A cryptographic hash function can be used to construct a **pseudorandom function (PRF)** or a **pseudorandom number generator (PRNG)**. A common application for a hash-based PRF is for the generation of symmetric keys.

Requirements And Security

For a hash value *h* = H(*x*), we say that *x* is the **preimage** of *h*. That is, *x* is a data block whose hash function, using the function H, is *h*. Because H is a many-to-one mapping, for any given hash value *h*, there will in general be multiple preimages.

A **collision** occurs if we have $x \neq y$ and H(*x*) = H(*y*). Because we are using hash functions for data integrity, collisions are clearly undesirable.

Table 11.1 Requirements for a Cryptographic Hash Function H

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness.

The fourth property, **preimage resistant**, is the one-way property: it is easy to generate a code given a message, but virtually impossible to generate a message given a code.

The fifth property, **second preimage resistant**, guarantees that it is impossible to find an alternative message with the same hash value as a given message.

A hash function that satisfies the first five properties in Table 11.1 is referred to as a weak hash function. If the sixth property, **collision resistant**, is also satisfied, then it is referred to as a strong hash function.

DIGITAL SIGNATURES

Properties

The digital signature must have the following properties:

- It must verify the author and the date and time of the signature.
- It must authenticate the contents at the time of the signature.
- It must be verifiable by third parties, to resolve disputes.

Thus, the digital signature function includes the authentication function.

Digital Signature Requirements

We can formulate the following requirements for a digital signature.

- The signature must be a bit pattern that depends on the message being signed.
- The signature must use some information unique to the sender to prevent both forgery and denial.
- It must be relatively easy to produce the digital signature.
- It must be relatively easy to recognize and verify the digital signature.
- It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
- It must be practical to retain a copy of the digital signature in storage.

The term **direct digital signature** refers to a digital signature scheme that involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source.

PUBLIC KEY INFRASTRUCTURE (PKI)

A *public key infrastructure (PKI)* is a set of roles, policies, and procedures needed to create, manage, distribute, use, store & revoke digital certificates and manage public-key encryption. The purpose of a PKI is to facilitate the secure electronic transfer of information for a range of network activities such as e-commerce, internet banking and confidential email.

In cryptography, a PKI is an arrangement that binds public keys with respective identities of entities (like people and organizations). The binding is established through a process of registration and issuance of certificates at and by a certificate authority (CA). Depending on the assurance level of the binding, this may be carried out by an automated process or under human supervision.

The PKI role that assures valid and correct registration is called a registration authority (RA). An RA is responsible for accepting requests for digital certificates and authenticating the entity making the request.

An entity must be uniquely identifiable within each CA domain on the basis of information about that entity.

A public key infrastructure (PKI) is a system for the creation, storage, and distribution of digital certificates which are used to verify that a particular public key belongs to a certain entity. The PKI creates digital certificates which map public keys to entities, securely stores these certificates in a central repository and revokes them if needed.

A PKI consists of:

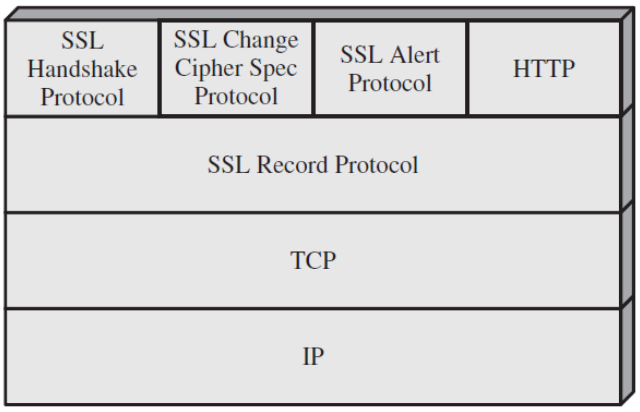
- A ***certificate authority (CA)*** that stores, issues and signs the digital certificates
- A ***registration authority (RA)*** which verifies the identity of entities requesting their digital certificates to be stored at the CA
- A ***central directory*** i.e., a secure location in which to store and index keys
- A ***certificate management system*** managing things like the access to stored certificates or the delivery of the certificates to be issued.
- A ***certificate policy*** stating the PKI's requirements concerning its procedures. Its purpose is to allow outsiders to analyze the PKI's trustworthiness.

SECURE SOCKETS LAYER

SSL is a general-purpose service implemented as a set of protocols that rely on TCP.

SSL Architecture

SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSL is not a single protocol but rather two layers of protocols, as illustrated in Figure.



The SSL Record Protocol provides basic security services to various higher layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of

SSL: the Handshake Protocol, The Change Cipher Spec Protocol, and the Alert Protocol. These SSL-specific protocols are used in the management of SSL exchanges.

➢ **SSL Record Protocol**

The SSL Record Protocol provides two services for SSL connections:

- **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
- **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

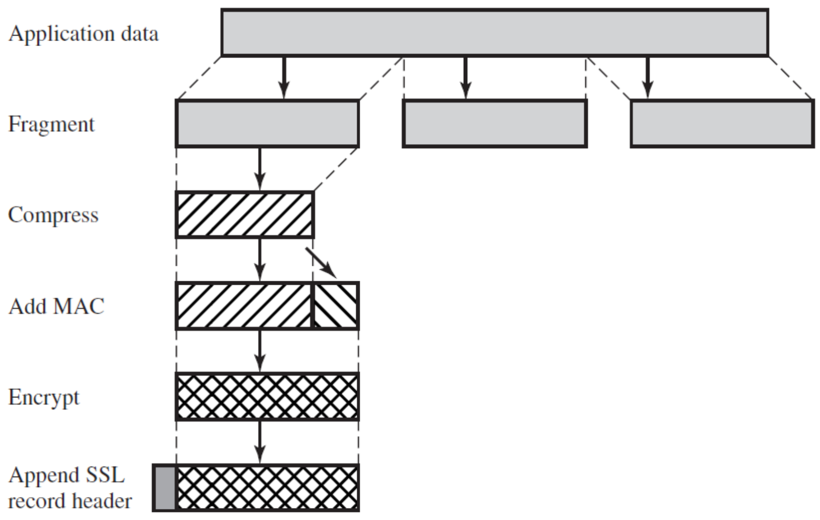


Figure 17.3 SSL Record Protocol Operation

The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled before being delivered to higher-level users.

The first step is **fragmentation**. Each upper-layer message is fragmented into blocks of 2^{14} bytes (16384 bytes) or less.

Next, **compression** is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes. In SSLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null.

The next step in processing is to compute a **message authentication code** over the compressed data.

Next, the compressed message plus the MAC are **encrypted** using symmetric encryption. Encryption may not increase the content length by more than 1024 bytes.

The final step of SSL Record Protocol processing is to prepare a header consisting of the following fields:

- **Content Type (8 bits):** The higher-layer protocol used to process the enclosed fragment.
- **Major Version (8 bits):** Indicates major version of SSL in use. For SSLv3, the value is 3.
- **Minor Version (8 bits):** Indicates minor version in use. For SSLv3, the value is 0.
- **Compressed Length (16 bits):** The length in bytes of the plaintext fragment (or compressed fragment if compression is used).

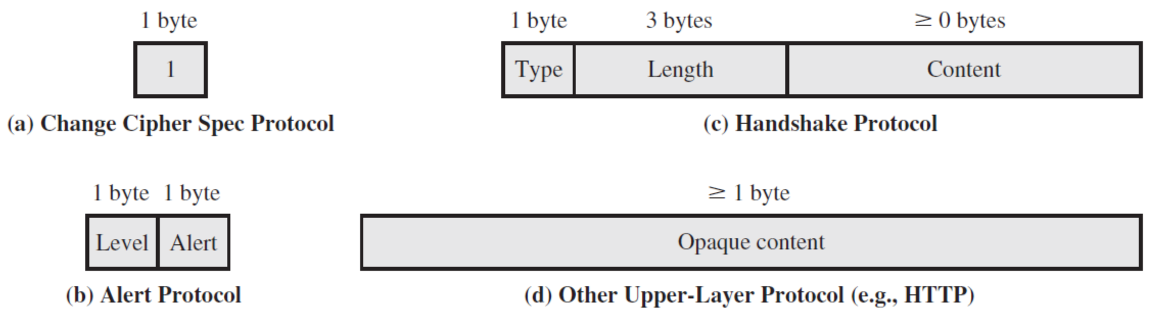


Figure 17.5 SSL Record Protocol Payload

➢ Change Cipher Spec Protocol

The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest. This protocol consists of a single message, which consists of a single byte with the value 1. The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.

➢ Alert Protocol

The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state.

Each message in this protocol consists of two bytes (Figure 17.5b). The first byte takes the value warning (1) or fatal (2) to convey the severity of the message. If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert.

➢ Handshake Protocol

The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The Handshake Protocol is used before any application data is transmitted. The Handshake Protocol consists of a series of messages exchanged by client and server. Each message has three fields:

- **Type (1 byte):** Indicates the type of message.
- **Length (3 bytes):** The length of the message in bytes.
- **Content (# 0 bytes):** The parameters associated with this message.

The exchange can be viewed as having four phases:

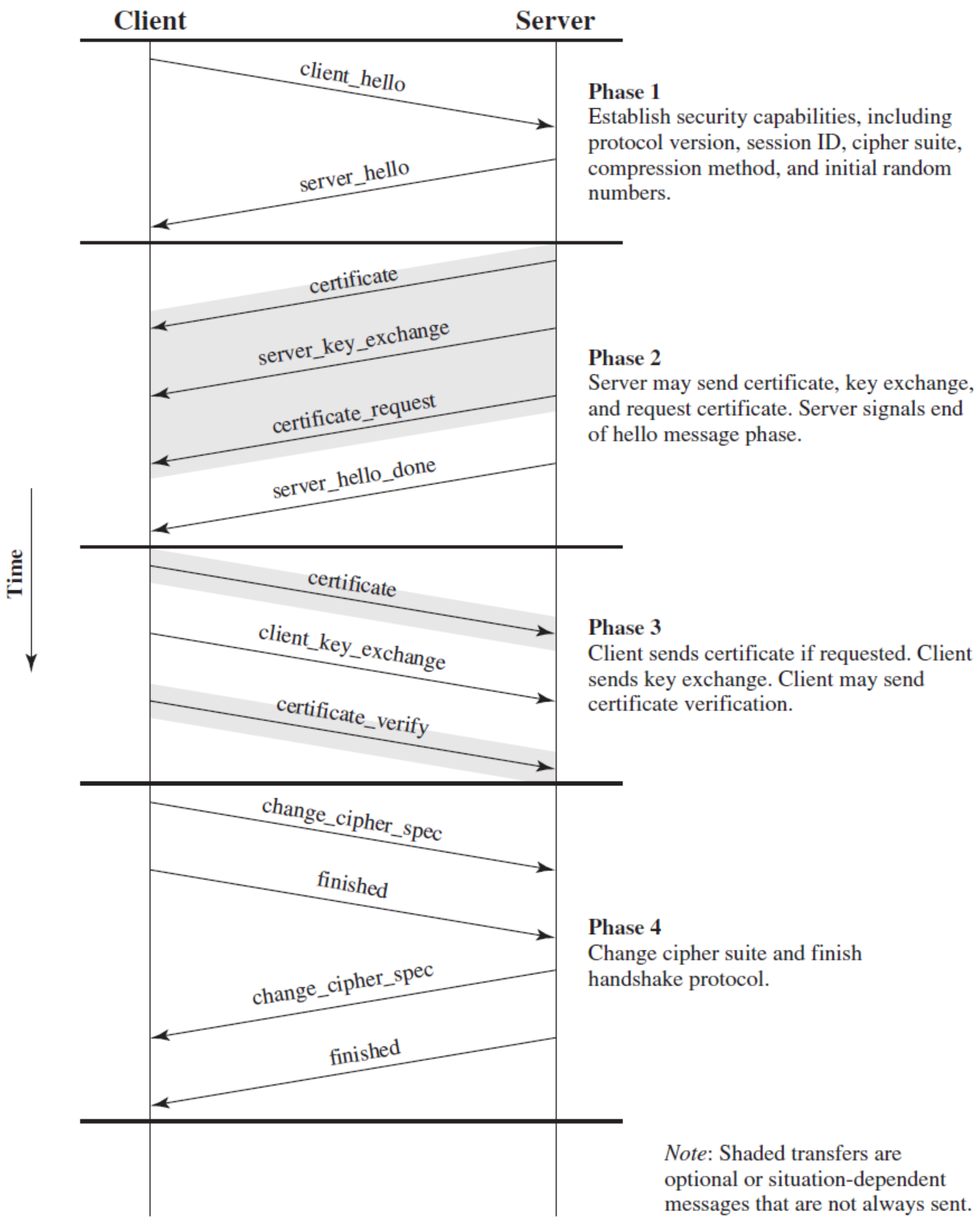

Client
Server
client_hello
server_hello
Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Phase 4
Change cipher suite and finish handshake protocol.
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

Figure 17.6 Handshake Protocol Action

HTTPS

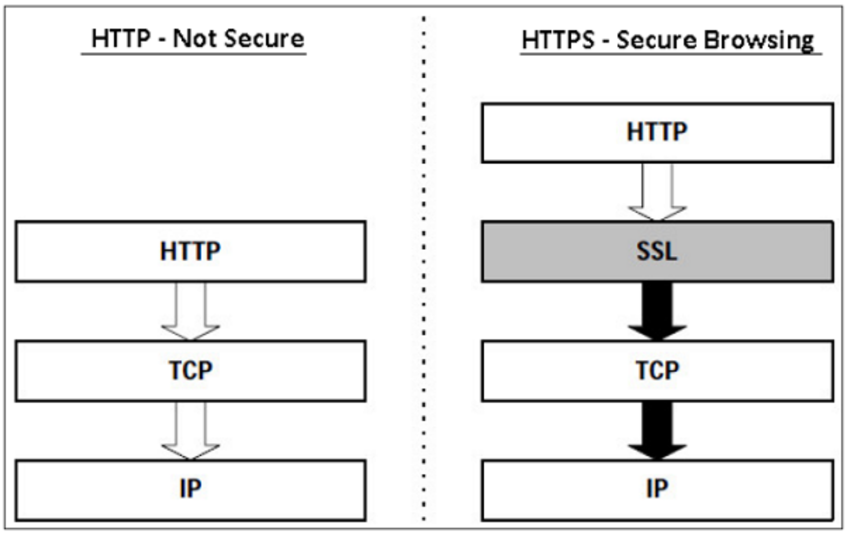
HTTPS (HTTP over SSL) refers to the combination of HTTP and SSL to implement secure communication between a Web browser and a Web server. The HTTPS capability is built into all modern Web browsers.

The principal difference seen by a user of a Web browser is that URL (uniform resource locator) addresses begin with https:// rather than http://. A normal HTTP connection uses port 80. If HTTPS is specified, port 443 is used, which invokes SSL.

When HTTPS is used, the following elements of the communication are encrypted:
- URL of the requested document
- Contents of the document

- Contents of browser forms (filled in by browser user)
- Cookies sent from browser to server and from server to browser
- Contents of HTTP header

There is no fundamental change in using HTTP over either SSL or TLS, and both implementations are referred to as HTTPS.



Working of HTTPS
HTTPS application protocol typically uses one of two popular transport layer security protocols - SSL or TLS. The process of secure browsing is described in the following points:

- We request a HTTPS connection to a webpage by entering https:// followed by URL in the browser address bar.

- Web browser initiates a connection to the web server. Use of https invokes the use of SSL protocol.

- An application, browser in this case, uses the system port 443 instead of port 80 (used in case of http).

- The SSL protocol goes through a handshake protocol for establishing a secure session.

- The website initially sends its SSL Digital certificate to the web browser. On verification of certificate, the SSL handshake progresses to exchange the shared secrets for the session.

- Once established, this session consists of many secure connections between the web server and the browser.

Use of HTTPS
- Use of HTTPS provides confidentiality, server authentication and message integrity to the user. It enables safe conduct of e-commerce on the Internet.

- Prevents data from eavesdropping and denies identity theft which are common attacks on HTTP.

Present day web browsers and web servers are equipped with HTTPS support. The use of HTTPS over HTTP, however, requires more computing power at the client and the server end to carry out encryption and SSL handshake.

INTRUSION DETECTION

Intrusion detection system (IDS)
An intrusion detection system (IDS) is a device or software application that monitors a network or systems for malicious activity or policy violations.

Any malicious activity or violation is typically reported either to an administrator or collected centrally using a security information and event management (SIEM) system.

A SIEM system combines outputs from multiple sources, and uses alarm filtering techniques to distinguish malicious activity from false alarms.

Intrusion prevention systems (IPS)
Intrusion prevention systems (IPS), also known as intrusion detection and prevention systems (IDPS), are network security appliances that monitor network or system activities for malicious activity. The main functions of intrusion prevention systems are to identify malicious activity, log information about this activity, report it and attempt to block or stop it.

Intrusion prevention systems are considered extensions of intrusion detection systems because they both monitor network traffic and/or system activities for malicious activity. The main differences are, unlike intrusion detection systems, intrusion prevention systems are to actively prevent or block intrusions that are detected.

Classification
IDS can be classified by where detection takes place (network or host) and the detection method that is employed (signature or anomaly-based).

Analyzed activity

- **Network intrusion detection systems**

A system that analyzes incoming network traffic is an example of an NIDS. Network intrusion detection systems (NIDS) are placed at a strategic point or points within the network to monitor traffic to and from all devices on the network.

It performs an analysis of passing traffic on the entire subnet, and matches the traffic that is passed on the subnets to the library of known attacks. Once an attack is identified, or abnormal behavior is sensed, the alert can be sent to the administrator.

- **Host intrusion detection systems**

A system that monitors important operating system files is an example of an HIDS. Host intrusion detection systems (HIDS) run on individual hosts or devices on the network. A HIDS monitors the inbound and outbound packets from the device only and will alert the user or administrator if suspicious activity is detected.

It takes a snapshot of existing system files and matches it to the previous snapshot. If the critical system files were modified or deleted, an alert is sent to the administrator to investigate.

Detection method

- **Signature-based**

Signature-based IDS refers to the detection of attacks by looking for specific patterns, such as byte sequences in network traffic, or known malicious instruction sequences used by malware.

- **Anomaly-based**

Anomaly-based intrusion detection systems were primarily introduced to detect unknown attacks, in part due to the rapid development of malware. The basic approach is to use machine learning to create a model of trustworthy activity, and then compare new behavior against this model.

Since these models can be trained according to the applications and hardware configurations, machine learning based method has a better generalized property in comparison to traditional signature-based IDS. Although this approach enables the detection of previously unknown attacks, it may suffer from false positives: previously unknown legitimate activity may also be classified as malicious.

Types
Intrusion detection systems can be classified into four different types:

- **Network-based:** monitors the entire network for suspicious traffic by analyzing protocol activity.
- **Wireless:** monitor a wireless network for suspicious traffic by analyzing wireless networking protocols.
- **Network-behavior analysis:** examines network traffic to identify threats that generate unusual traffic flows, such as distributed denial of service (DDoS) attacks, certain forms of malware and policy violations.
- **Host-based:** an installed software package which monitors a single host for suspicious activity by analyzing events occurring within that host.

MALICIOUS PROGRAMS

Trojan horse
A Trojan horse is a program that allows the attacker to control the user's computer from a remote location. The program is usually disguised as something that is useful to the user. Once the user has installed the program, it has the ability to install malicious payloads, create backdoors, install other unwanted applications that can be used to compromise the user's computer, etc.

The list below shows some of the activities that the attacker can perform using a Trojan horse:

- Use the user's computer as part of the Botnet when performing distributed denial of service attacks.
- Damage the user's computer
- Stealing sensitive data such as stored passwords, credit card information, etc.
- Modifying files on the user's computer
- Electronic money theft by performing unauthorized money transfer transactions
- Log all the keys that a user presses on the keyboard and sending the data to the attacker. This method is used to harvest user ids, passwords, and other sensitive data.
- Viewing the users' screenshot
- Downloading browsing history data

Worm
A worm is a malicious computer program that replicates itself, attaching themselves to different files and looking for pathways between computers, such as computer network that shares common file storage areas.

An attacker may use a worm to accomplish the following tasks:

- Install backdoors on the victim's computers. The created backdoor may be used to create zombie computers that are used to send spam emails, perform distributed denial of service attacks, etc. the backdoors can also be exploited by other malware.
- Worms may also slowdown the network by consuming the bandwidth as they replicate.
- Install harmful payload code carried within the worm.

Ransomware
Ransom malware, or ransomware, is a type of malware that prevents users from accessing their system or personal files and demands ransom payment in order to regain access. Ransomware encrypts data in the computer with a key which is unknown to the user. The user has to pay a ransom (price) to the hacker to retrieve data. Once the amount is paid the victim can resume using his/her system.

Virus
A virus is a fragment of code embedded in a legitimate program. Virus are self-replicating and are designed to infect other programs. They can wreak havoc in a system by modifying or destroying files

causing system crashes and program malfunctions. Viruses can consume computer resources such as memory and CPU time. The attacked programs and files are said to be "infected".

A computer virus may be used to:

- Access private data such as user id and passwords
- Display annoying messages to the user
- Corrupt data in your computer
- Log the user's keystrokes

Computer viruses have been known to employ social engineering techniques. These techniques involve deceiving the users to open the files which appear to be normal files such as Word or Excel documents. Once the file is opened, the virus code is executed and does what it's intended to do.

Various types of virus:

File Virus: This type of virus infects the system by appending itself to the end of a file. It changes the start of a program so that the control jumps to its code. After the execution of its code, the control returns back to the main program. Its execution is not even noticed.

Boot sector Virus: It infects the boot sector of the system, executing every time system is booted and before operating system is loaded. It infects other bootable media like floppy disks. These are also known as memory virus as they do not infect file system.

Macro Virus: Unlike most virus which are written in low-level language(like C or assembly language), these are written in high-level language like Visual Basic. These viruses are triggered when a program capable of executing a macro is run.

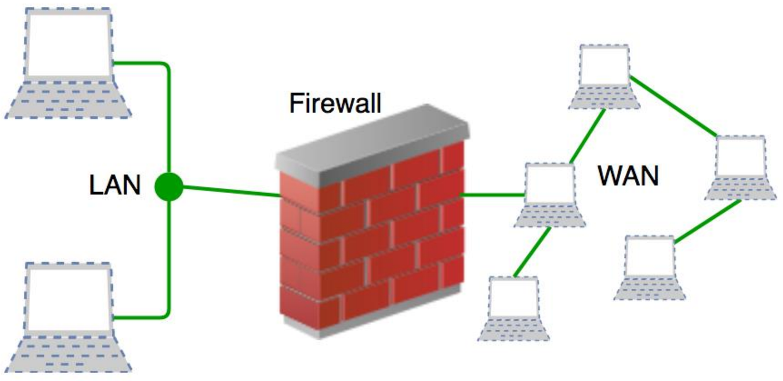
Polymorphic Virus: A virus signature is a pattern that can identify a virus(a series of bytes that make up virus code). So in order to avoid detection by antivirus a polymorphic virus changes each time it is installed. The functionality of virus remains same but its signature is changed.

Encrypted Virus: In order to avoid detection by antivirus, this type of virus exists in encrypted form. It carries a decryption algorithm along with it. So the virus first decrypts and then executes.

Stealth Virus: It is a very tricky virus as it changes the code that can be used to detect it. Hence, the detection of virus becomes very difficult. For example, it can change the read system call such that whenever user asks to read a code modified by virus, the original form of code is shown rather than infected code.

FIREWALL

In computing, a firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted internal network and untrusted external network, such as the Internet.



Before Firewalls, network security was performed by *Access Control Lists (ACLs)* residing on routers. ACLs are rules that determine whether network access should be granted or denied to specific IP address.

But ACLs cannot determine the nature of packet it is blocking. Also, ACL alone does not have the capacity to keep threats out of the network. Hence, Firewall was introduced.

Types
Firewalls are generally categorized as network-based or host-based:

- **Network-based firewalls**

Network-based firewalls filter traffic between two or more networks. They are positioned on the gateway computers of LANs, WANs and intranets. They are either software appliances running on general-purpose hardware, or hardware-based firewall computer appliances. Firewall appliances may also offer other functionality to the internal network they protect, such as acting as a DHCP or VPN server for that network.

- **Host-based firewalls**

Host-based firewalls run on host computers and control network traffic in and out of those machines. They are positioned on the network node itself to control network traffic in and out of those machines.