IPSec

IP version 4.0 doesn’t include a native security method, so IPSec was created. Not only is IPSec considered an addition to IPv4, it also has been incorporated into the IPv6 protocol.

Based on cryptography, Internet Protocol Security, or IPSec, ensures the privacy of network traffic as well as its authentication. IPSec is used for peer-to-peer and client-server communications across a private or public network; secure LAN-to-LAN communications across a WAN; and remote access transmissions via either dialup or virtual private network (VPN). IPSec functions at the Network layer. Upper-layer protocols utilize IPSec to provide secure transport of application data. The fact that IPSec is a network-layer protocol makes its services transparent to applications. A user doesn’t need to invoke any special security software to use IPSec. Instead, the administrator configures IPSec, and those services are provided to IP datagrams according to the configuration.

IPSec provides several types of services that prevent different kinds of attacks from interfering with the network. These include authentication and encryption of the data stream. While other security strategies are built to create a perimeter of privacy around a network, IPSec ensures that data cannot be tampered with while it is traversing any part of the network—whether it’s within the private network boundaries or across the public Internet.

The inner workings of IPSec, as defined by the Internet Engineering Task Force (IETF), are based on its authentication header (AH) and encapsulated security payload (ESP). As you can probably guess, AH provides authentication for data integrity and covers the entire packet, while ESP provides encryption for confidentiality. IPSec uses User Datagram Protocol (UDP) port 500. When you use IPSec across a firewall, you must ensure that UDP port 500 is open, along with IP type 50, which is used by ESP, and IP type 51, which is used by AH. Because ESP provides encryption, it negatively affects the size of the packet. To reduce the size of the packet, IPSec includes IP payload compression (IPcomp), which compresses the packet before it is encrypted by ESP.

Internet Key Exchange (IKE) is the part of IPSec that provides the key. A security key is agreed upon using the Diffie-Hellman Technique is and shared by the sender and recipient, ensuring that data is not changed during transmission. The Diffie-Hellman Technique is a public key cryptography algorithm, which is a fancy name for an equation, that enables two computers to communicate and agree upon a shared key. The process begins with each of the computers exchanging their public key. Then each one combines its private information along with the other’s public information to generate a shared secret value.

An IPSec session is initiated when the IP protocol receives data from upper layers. IPSec works with the destination computer to agree upon the shared key and then encrypts the data at the sending host. This process ensures that the data packets are unreadable while en route to their destination, where they are then decrypted using the shared key.

IPSec supports two types of encryption modes—transport and tunnel. The transport mode encrypts only the data part of the packet, not the header. Tunnel mode encrypts the entire header and data.

When data is sent using IPSec, the AH follows the standard IP header. After AH, the ESP header is next. These headers are added before the transport layer headers, whether those are UDP or TCP, which means that the transport-layer header is encrypted.

Secure Sockets Layer (SSL)

SSL is the abbreviation for Secure Sockets Layer, but users will likely be more familiar with its manifestation as the HTTPS:// that precedes the URL of a secure Web site. SSL is a protocol that uses a public key to encrypt the data transmitted across the Internet. It is commonly used to provide privacy for sensitive information such as credit card numbers.

SSL runs transparently to applications, because it sits below upper-layer applications and above the IP protocol, as shown in Figure 8-2.

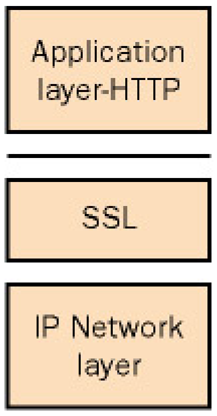


Figure 8-2: SSL is transparent to users because it works below the application layer.

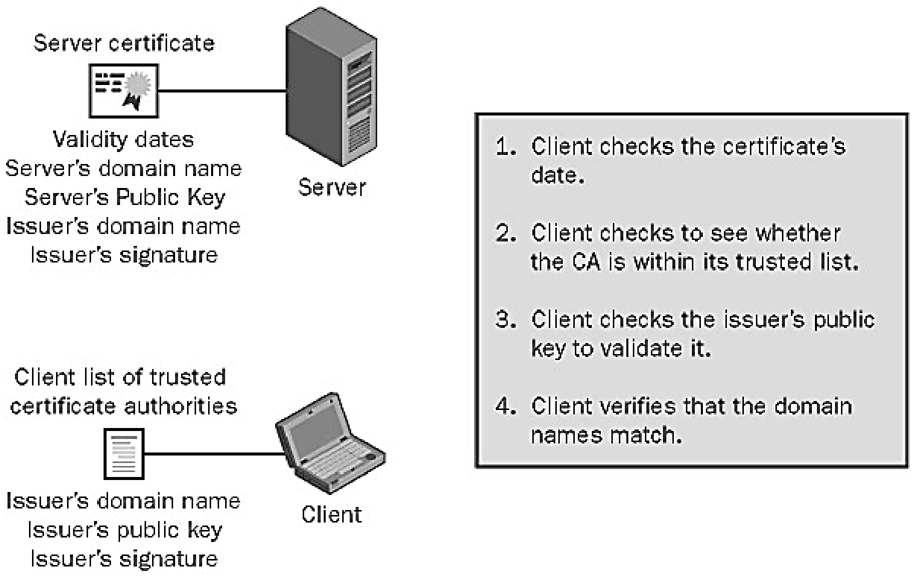
Working on behalf of upper-layer protocols, the SSL server authenticates itself using a certificate and public ID to an SSL-enabled client, which includes both Netscape Navigator and Microsoft Internet Explorer Web browsers, and others. The SSL client ensures that the server's certificate has been issued by a trusted certificate authority (CA), it authenticates itself back to the server using the same process, and an encrypted link is created between the two. A CA is a server that issues certificates to validate hosts within certain domains. This process ensures that both the client's and the server's identities are confirmed. During the ensuing data transmission, SSL enacts a mechanism to ensure that the data is not tampered with before it reaches its destination. The two subprotocols within SSL enable this entire process. The SSL handshake protocol exchanges the series of messages at the initiation of an SSL connection to establish a link. The SSL record protocol specifies the format that will be used to transmit the data.

SSL is able to use several different types of ciphers:

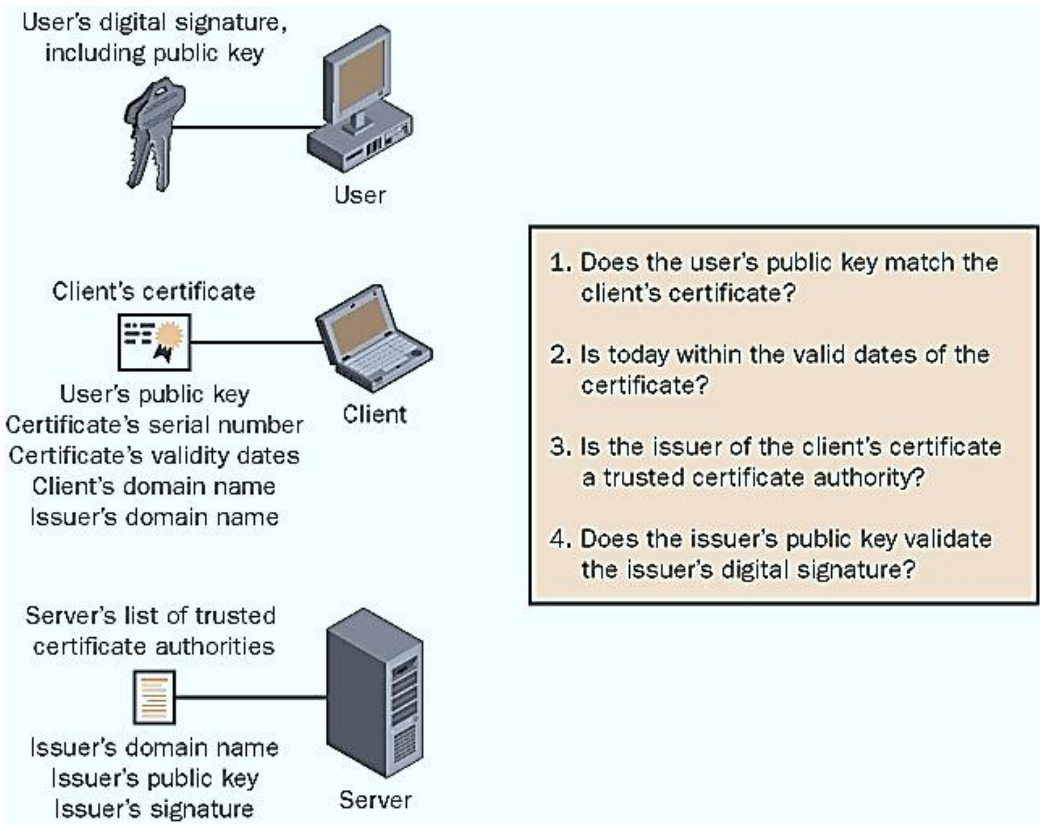
- **Data encryption standard (DES) and Triple DES.** DES is a private key exchange that applies a 56-bit key to each 64-bit block of data. Triple DES is the application of three DES keys in succession.
- **Key Exchange Algorithm (KEA).** KEA enables the client and server to establish mutual keys to use in encryption.
- **Message Digest version 5 (MD5).** This cipher creates a 128-bit message digest to validate data.
- **Rivest-Shamir-Adleman (RSA).** This is the most commonly used key exchange for SSL. It works by multiplying two large prime numbers, and through an algorithm determining both public and private keys. The private key does not need to be transmitted across the Internet but is able to decrypt the data transmitted with the public key.
- **Secure Hash Algorithm (SHA).** SHA produces a message digest of 160 bits using the SHA-1 80-bit key to authenticate the message.

A network administrator can enable or disable ciphers for the network. When an SSL client initiates contact with an SSL server, the SSL handshake protocol is used to agree upon the cipher to use. They identify the strongest ciphers that the two share, and then use those for the session. In general, the stronger the cipher used, the slower the transmission. An admin should balance the need for security with the transmission speed required, as well as the international export laws regarding which ciphers can be used across country boundaries.

When a client establishes the trustworthiness of an SSL server, it uses the server's public key, the certificate's (delivered by the CA) serial number and validity period, the server's domain name and the CA's domain name, and finally the CA's digital signature. The steps that a client uses to validate an SSL server are shown in Figure 8-3.


Server certificate
Validity dates
Server's domain name
Server's Public Key
Issuer's domain name
Issuer's signature
Server
Client list of trusted certificate authorities
Issuer's domain name
Issuer's public key
Issuer's signature
Client
1. Client checks the certificate's date.
2. Client checks to see whether the CA is within its trusted list.
3. Client checks the issuer's public key to validate it.
4. Client verifies that the domain names match.
Figure 8-3: A client makes certain that the SSL server's certificate is issued by a trusted CA.

The server goes through a similar process, usually requiring the same type of information from a client during the authentication of the SSL client. This process is depicted in Figure 8-4.


User's digital signature, including public key
User
Client's certificate
User's public key
Certificate's serial number
Certificate's validity dates
Client's domain name
Issuer's domain name
Client
Server's list of trusted certificate authorities
Issuer's domain name
Issuer's public key
Issuer's signature
Server
1. Does the user's public key match the client's certificate?
2. Is today within the valid dates of the certificate?
3. Is the issuer of the client's certificate a trusted certificate authority?
4. Does the issuer's public key validate the issuer's digital signature?
Figure 8-4: Clients are authenticated by SSL servers.

Kerberos

Kerberos was the name of the legendary three-headed dog who guarded the gates of Hades. It is an appropriate name for the enormously strong security protocol developed by the Athena project at the Massachusetts Institute of Technology (MIT). As the guard dog of information transmitted across the Internet, Kerberos is an authentication protocol that is used to establish trust relationships between domains and verify the identities of users and network services.

When an entity attempts to access a Kerberos-protected resource and provides correct authentication information, Kerberos issues a *ticket* to it. This method does not require a password for transmission across the network. The ticket is actually a temporary certificate with the information required to identify the entity to the network. The entity uses this Kerberos ticket to request further Kerberos tickets to allow it to access subsequent services on the network. Each process requires a complex mutual authentication, but this is completely transparent to the user.

Specifically, the first request for access to the network is passed to an Authentication Server (AS). The AS creates a ticket-granting ticket, which is an encryption key based on the password, the user name, and a value that represents the requested service.

The ticket-granting ticket must then be sent to a ticket-granting server. This ticket-granting server sends back a new time-stamped Kerberos ticket, which can then be sent to the server requesting a specific service. The final server either accepts the ticket and provides the service or rejects it. The time stamp on the Kerberos ticket enables it to be used only for a limited time.

Because of the use of time-stamping, computers using Kerberos authentication require fairly tightly synchronized time settings. Given the unreliability of computer time clocks, the best way to ensure that computers are synchronized is to use a network time service.

Kerberos Trust Relationships

Kerberos enables trusts to be established between two different UNIX realms, between two Windows 2000 domains, or even between a UNIX realm and a Windows 2000 domain. Trust relationships are established using Kerberos so that authentication credentials can be passed on to network resources in trusted domains or realms. For example, when domain A trusts domain B, the users in domain B are able to access resources in domain A.

Kerberos trust relationships are typically *transitive* and *bidirectional* in nature. Transitive means that if domain A trusts domain B, and domain B trusts domain C, domain A is understood to trust domain C. Bidirectional means that when domain A trusts domain B, domain B automatically trusts domain A. The only time a trust relationship is nontransitive and unidirectional is in the case of connection between a UNIX realm and a Windows 2000 domain, between two UNIX realms, or between domains within two different Windows 2000 Active Directory forests. (A *forest* is a group of related domains considered to be a single Active Directory entity.)

Let's look at how a trust relationship would work in an Active Directory forest of domains. Figure 8-5 shows a forest in which the root domain is named rootdomain.com. There are two subdomains, named one.rootdomain.com and two.rootdomain.com. Rootdomain.com has a Kerberos trust with each of its subdomains. Because it is bidirectional, the subdomains trust the rootdomain.com in return. Because those Kerberos trusts are transitive, it is understood that one.rootdomain.com trusts two.rootdomain.com and vice versa.

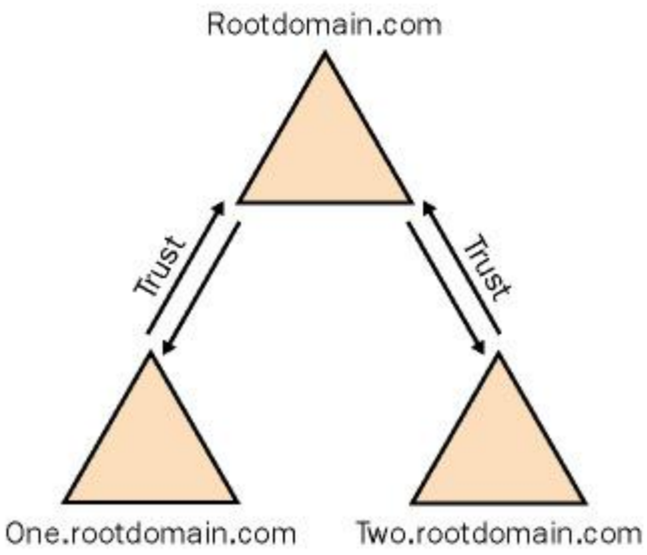
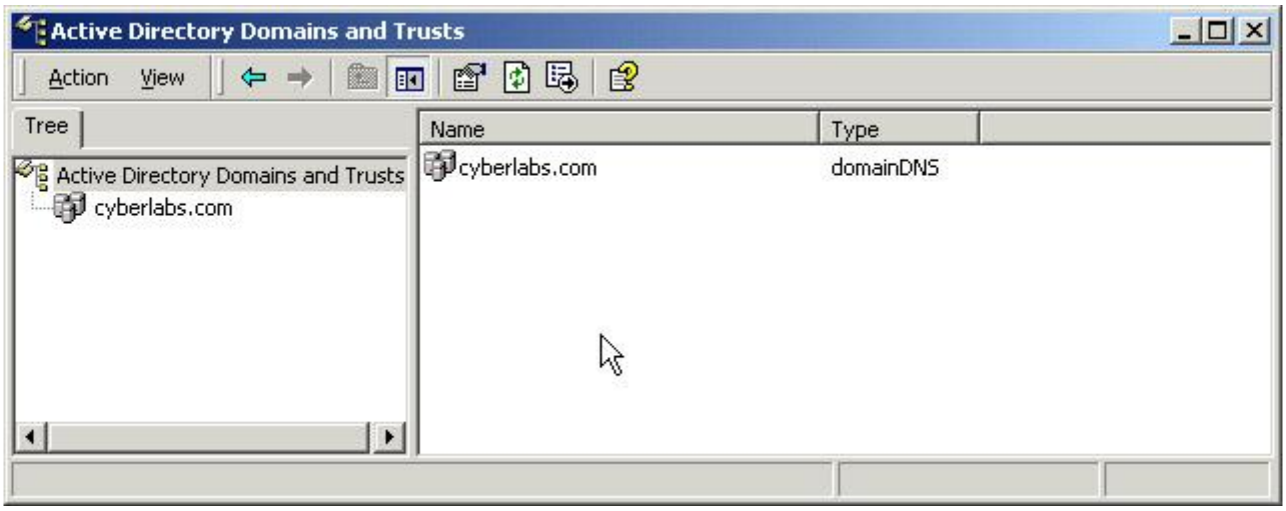


Figure 8-5: An Active Directory forest is related through transitive, bidirectional Kerberos trusts.

Wherever a Kerberos trust exists, the users in one domain will be able to access resources in the other domain as long as the administrator has granted those users access. A network administrator can view Kerberos trust relationships in the Windows 2000 Active Directory Domains and Trusts console, shown in Figure 8-6. To access this console, click the Start menu and point to Programs and then Administrative Tools. Click Active Directory Domains And Trusts On A Windows 2000 Domain Controller.



Active Directory Domains and Trusts
cyberlabs.com

Name	Type
cyberlabs.com	domainDNS

Figure 8-6: The Active Directory Domains and Trusts console shows Kerberos trusts.

How Kerberos works?

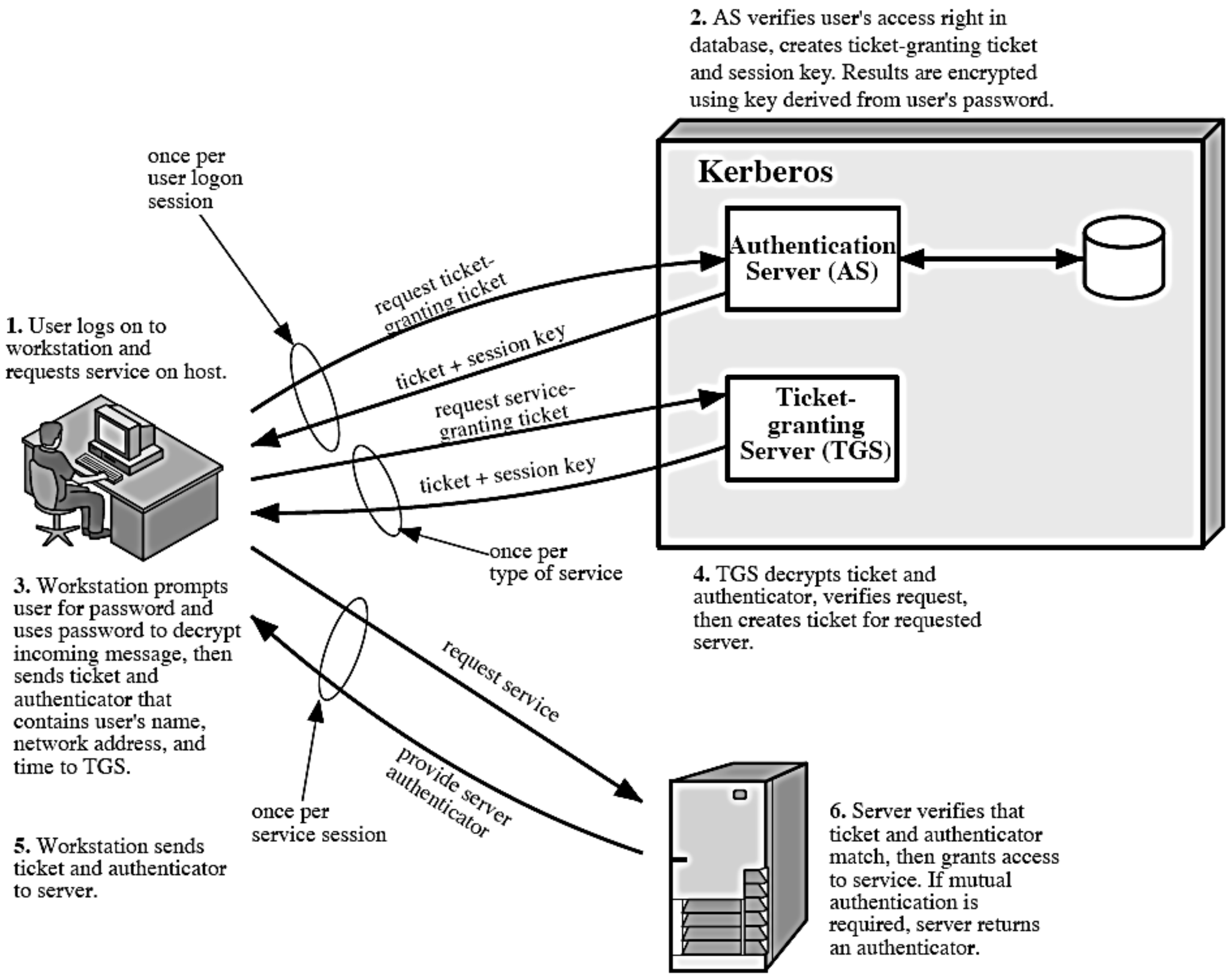

2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
Kerberos
Authentication Server (AS)
Ticket-granting Server (TGS)
once per user logon session
1. User logs on to workstation and requests service on host.
request ticket-granting ticket
ticket + session key
request service-granting ticket
ticket + session key
once per type of service
3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.
4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.
request service
provide server authenticator
once per service session
5. Workstation sends ticket and authenticator to server.
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Figure 14.1 Overview of Kerberos

Firewalls

When a network administrator initially establishes the perimeter of defense for a network, one of the first pieces of equipment installed is a firewall. This piece of equipment is actually a router with two interfaces—one leading to the public network and the other to the private network. Keep in mind that the “public” network doesn’t need to be the Internet—it can be a regular network—while the “private” network can be a portion of the network where the most sensitive data is stored or used. For example, a large software development company that often uses contract labor might decide to place all research and development information within a private network that is bounded by firewalls from the rest of the network.

The simple status of router does not qualify a piece of equipment to serve as a firewall. One of the methods a firewall uses to secure the network is *packet filtering*. This is the process of receiving data packets from one interface and examining them to see which packets meet the rules. For packets that meet firewall rules, they are either permitted or blocked, depending on how the rule is implemented. For example, a person sets up a firewall and implements a rule to block all traffic for Telnet (TCP port 23) coming from the public network. All other traffic can flow through the firewall. A different way of setting up that firewall is to block all traffic and then permit only the traffic that you want to receive. There are other criteria upon which to permit or deny traffic; you can decide that the traffic shouldn’t be received from a certain IP address (or sent to a certain IP address) or that a type of traffic should be denied. This can come in handy if a network administrator decides to block all e-mail traffic from a domain where a lot of spam e-mail messages originate.

Whether you block a certain type of traffic and allow the rest, or block all traffic and allow a set of traffic types can make a major difference in how a data packet is handled. For example, if you’ve decided to block all Telnet traffic on TCP port 23 in an attempt to stop denial-of-service attacks because some of them use Telnet, you won’t be successful; a Telnet request can be configured to use a different port. In addition, other methods can use other ports and achieve the very same result—a network outage. On the other hand, if you decide to block all traffic and allow only certain types, you should really understand which data to allow and which not to. Some applications will function only across a specialized port, and using such a policy on the firewall can cause them to fail. However, this is the more secure method of implementing firewall rules.

Firewalls are useful for protecting the network from unauthorized access to data, as well as for protection against denial-of-service attacks, but they’re not foolproof. If you need to use a particular service, that type of traffic must be permitted to pass through. For each packet type permitted to enter the network, there is a risk. However, because the firewall acts as a choke point—the only access in or out of the network—it’s easy to use it to log access attempts.

You should know that firewalls can’t protect data that doesn’t pass through the network. Several years ago, I sent a team to consult with a company that had been victim to unauthorized access to data—including sensitive information that somehow ended up in the local news. A thorough review of the network’s security finally revealed that the company’s data was accessible to the public because an executive secretary was putting sensitive reports into the garbage can. This information was available to anyone who could sift through a pile of papers. A consistent security policy must be established and adhered to for the entire company, not just the network.

Before configuring a firewall, you should first decide what type of traffic should and what type shouldn’t be allowed to pass through it. Figure 8-7 depicts a scenario in which DNS, HTTP, and SMTP traffic is allowed into the network, any other incoming traffic is denied, and all outgoing traffic is allowed. It’s always a good idea to end an access control list on a firewall with a deny to all other traffic because this will prevent any attacks through higher ports—which can lead to a software vulnerability or even an intentionally placed back door into an application.

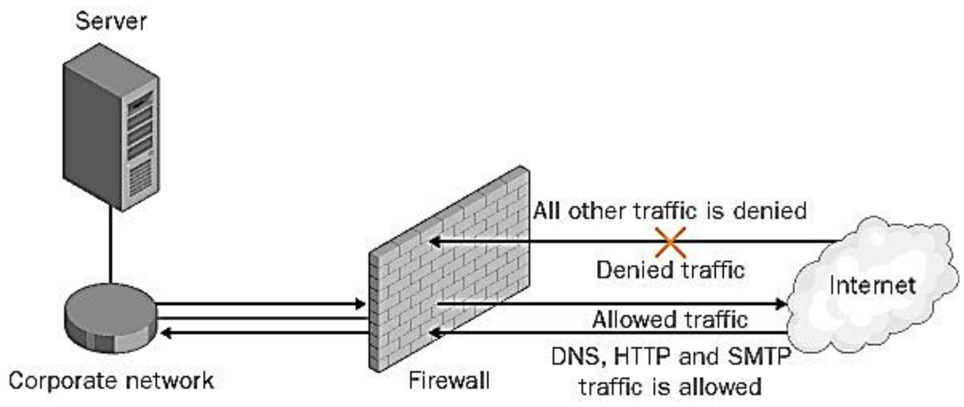


Figure 8-7: A firewall is usually more restrictive to incoming traffic than outgoing traffic.

A firewall uses an access control list for all the commands to execute packet filters. When the firewall implements the access control list, it does so in order of the commands. This means that when a “deny all” command precedes a “permit this” command, the “permit” command is not executed. This process can cause problems in application performance.

Another reason an application might not perform correctly after a firewall is installed is the port number used. Many applications use one or more TCP or UDP ports that are not well known. When implementing a new firewall, you should review every application that must function across the firewall. When the port numbers are known, they must be permitted within the access control list on the firewall.

Demilitarized Zones

A demilitarized zone (DMZ) is an offshoot from a firewall that is not considered part of the Internet, nor is it considered part of the private network. As shown in Figure 8-8, a DMZ is a middle area that offers more freedom of access from the Internet. This configuration places the DMZ between two firewalls.

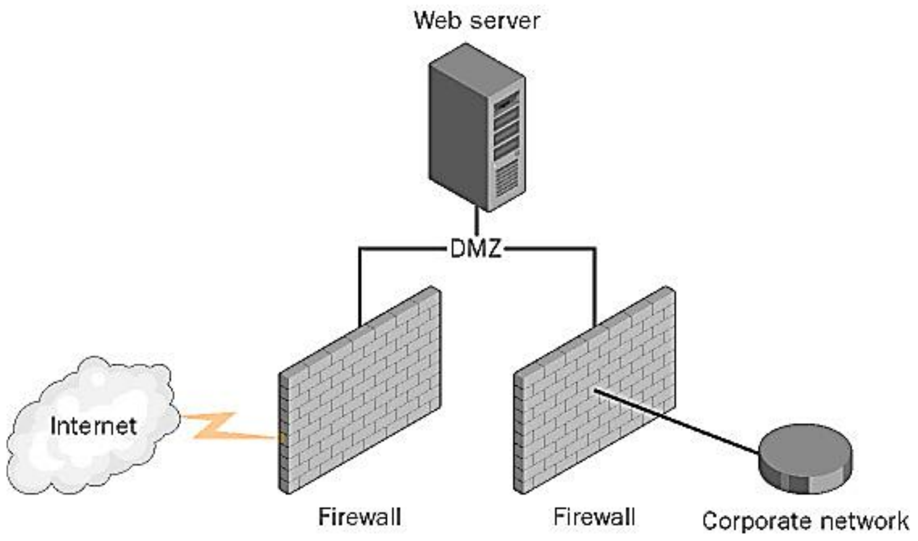


Figure 8-8: The DMZ can sit between a firewall to the Internet and a firewall to the private network.

Alternatively, a DMZ can be an offshoot area such as that displayed in Figure 8-9. In this scenario, the firewall has three interfaces, one that connects to the Internet, a second that connects to the DMZ, and a third that connects to the private network. This configuration is driven solely by access control lists in which the DMZ access is relaxed compared to that of the private network.

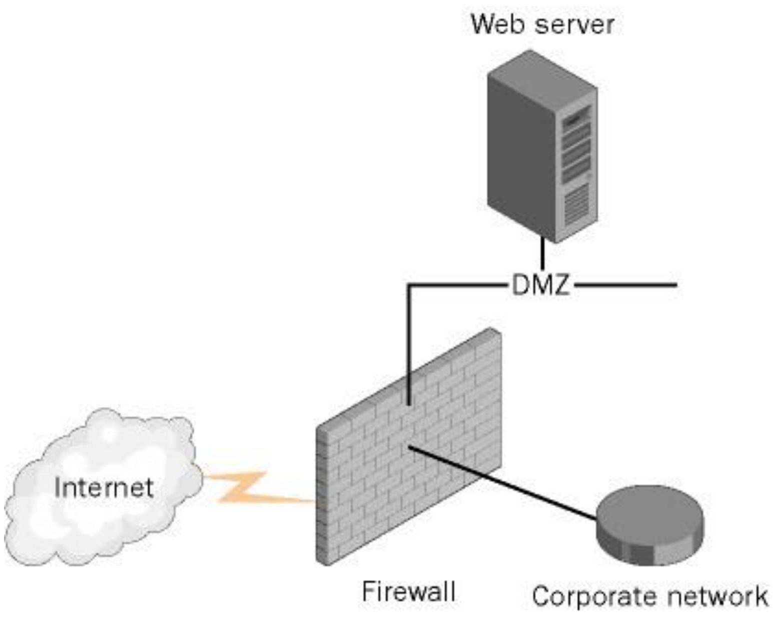


Figure 8-9: The DMZ can be configured as a third interface on a firewall.

One reason to create a DMZ is to provide access to certain servers, such as a Web server or e-mail server, yet still protect the rest of the network from those types of traffic.

Source: MSPress – 'CompTIA Network+ - Faster Smarter Network Certific'