APPENDIX A

Understanding TCP/IP

Introduction

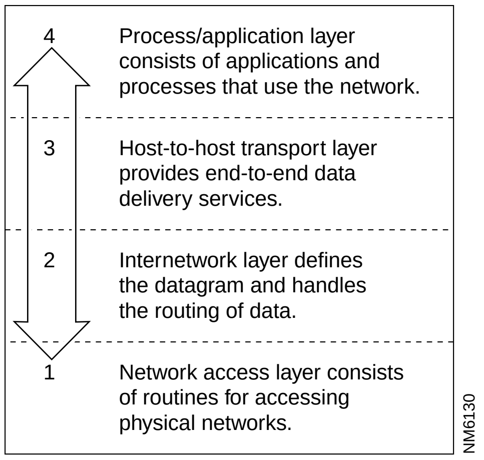
To fully understand the architecture of Cisco Centri Firewall, you need to understand the TCP/IP architecture on which the Internet is based. This appendix discusses the TCP/IP architecture and provides a basic reference model that can help you understand how Cisco Centri Firewall operates. It explains TCP/IP terminology and describes the fundamental concepts underlying the TCP/IP protocol suite. We begin by providing a common frame of reference to use as a basis for the rest of the discussion contained in this appendix on TCP/IP and Cisco Centri Firewall.

What is an Architectural Model?

An *architectural model* provides a common frame of reference for discussing Internet communications. It is used not only to explain communication protocols but to develop them as well. It separates the functions performed by communication protocols into manageable layers stacked on top of each other. Each layer in the stack performs a specific function in the process of communicating over a network.

Generally, TCP/IP is described using three to five functional layers. To describe TCP/IP based firewalls more precisely, we have chosen the common *DoD reference model*, which is also known as the *Internet reference model*. Figure A-1 illustrates the Internet reference model.

Figure A-1 The DoD Protocol Model


4
Process/application layer consists of applications and processes that use the network.
3
Host-to-host transport layer provides end-to-end data delivery services.
2
Internetwork layer defines the datagram and handles the routing of data.
1
Network access layer consists of routines for accessing physical networks.
NM6130

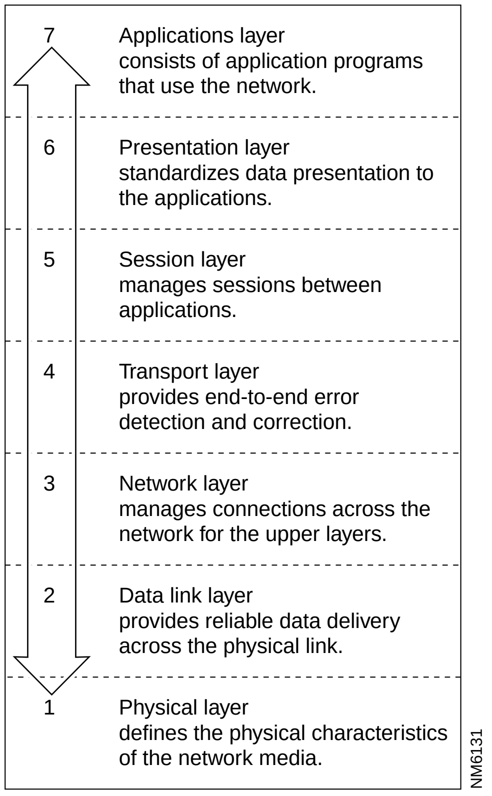
This model is based on the three layers defined for the DoD Protocol Model in the *DDN Protocol Handbook, Volume 1*. These three layers are as follows:

- network access layer
- host-to-host transport layer
- application layer

An additional layer, the internetwork layer, has been added to this model. The internetwork layer is commonly used to describe TCP/IP. The following section explains how network protocols work, and it defines the basic terminology that we use to discuss TCP/IP and Cisco Centri Firewall.

Another standard architectural model that is often used to describe a network protocol stack is the OSI reference model. This model consists of a seven layer protocol stack (see Figure A-2).

Figure A-2 The OSI Protocol Reference Model


7
Applications layer
consists of application programs
that use the network.
6
Presentation layer
standardizes data presentation to
the applications.
5
Session layer
manages sessions between
applications.
4
Transport layer
provides end-to-end error
detection and correction.
3
Network layer
manages connections across the
network for the upper layers.
2
Data link layer
provides reliable data delivery
across the physical link.
1
Physical layer
defines the physical characteristics
of the network media.
NM6131

No additional information or explanation for this reference model will be included within this guide because very few firewalls implement this model. For additional information on this reference model consult Chapman, D. B., and Elizabeth D. Zurichy, *Building Internet Firewalls*, Sebastopol:O'Reilly & Associates, Inc., September 1995. (See Appendix B.) or Heywood, D., *Networking with Microsoft TCP/IP*, Indianapolis: New Riders Publishing, 1996. (See Chapter 1.)

Understanding Architectural Models and Protocols

In an architectural model, a layer does not define a single protocol—it defines a data communication function that may be performed by any number of protocols. Because each layer defines a function, it can contain multiple protocols, each of which provides a service suitable to the function of that layer.

Every protocol communicates with its peer. A *peer* is an implementation of the same protocol in the equivalent layer on a remote computer. Peer-level communications are standardized to ensure that successful communications take place. Theoretically, each protocol is only concerned with communicating to its peer—it does not care about the layers above or below it.

A dependency, however, exists between the layers. Because every layer is involved in sending data from a local application to an equivalent remote application, the layers must agree on how to pass data between themselves on a single computer. The upper layers rely on the lower layers to transfer the data across the underlying network.

How a Protocol Stack Works

As the reference model indicates, protocols (which compose the various layers) are like a pile of building blocks stacked one upon another. Because of this structure, groups of related protocols are often called *stacks* or *protocol stacks*.

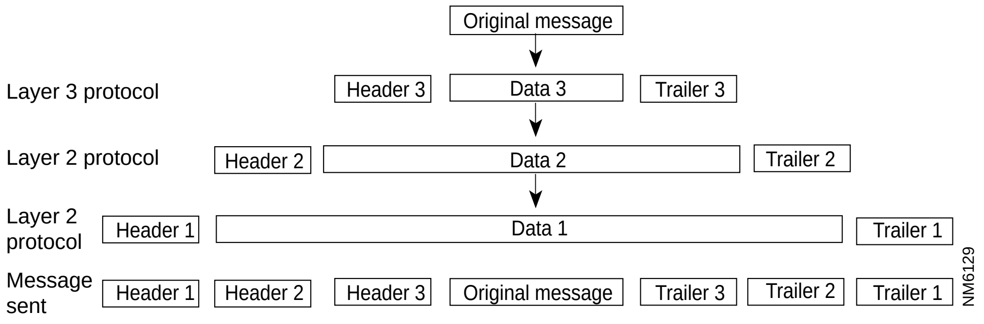
Data is passed down the stack from one layer to the next, until it is transmitted over the network by the network access layer protocols. The four layers in this reference model are crafted to distinguish between the different ways that the data is handled as it passes down the protocol stack from the application layer to the underlying physical network.

At the remote end, the data is passed up the stack to the receiving application. The individual layers do not need to know how the layers above or below them function; they only need to know how to pass data to them.

Each layer in the stack adds control information (such as destination address, routing controls, and checksum) to ensure proper delivery. This control information is called a *header* and/or a *trailer* because it is placed in front of or behind the data to be transmitted. Each layer treats all of the information that it receives from the layer above it as data, and it places its own header and/or trailer around that information.

These wrapped messages are then passed into the layer below along with additional control information, some of which may be forwarded or derived from the higher layer. By the time a message exits the system on a physical link (such as a wire), the original message is enveloped in multiple, nested wrappers—one for each layer of protocol through which the data passed. When a protocol uses headers or trailers to package the data from another protocol, the process is called *encapsulation.* This process is illustrated in Figure A-3.

Figure A-3 Encapsulation of Data for Network Delivery



When data is received, the opposite happens. Each layer strips off its header and/or trailer before passing the data up to the layer above. As information flows back up the stack, information received from a lower layer is interpreted as both a header/trailer and data. The process of removing headers and trailers from data is called *decapsulation*. This mechanism enables each layer in the transmitting computer to communicate with its corresponding layer in the receiving computer. Each layer in the transmitting computer communicates with its *peer* layer in the receiving computer via a process called *peer-to-peer communication*.

Each layer has specific responsibilities and specific rules for carrying out those responsibilities, and it knows nothing about the procedures that the other layers follow. A layer carries out its tasks and delivers the message to the next layer in the protocol stack. An *address mechanism* is the common element that allows data to be routed through the various layers until it reaches its destination.

Each layer also has its own independent data structures. Conceptually, a layer is unaware of the data structures used by the layers above and below it. In reality, the data structures of a layer are designed to be compatible with the structures used by the surrounding layers for the sake of more efficient data transmission. Still, each layer has its own data structures and its own terminology to describe those structures.

The following section describes the Internet reference model in more detail. We will use this reference model throughout this guide to describe the structure and function of the TCP/IP protocol suite and Cisco Centri Firewall.

Understanding the Internet Reference Model

As mentioned earlier, the Internet reference model contains four layers: the network access layer, the internetwork layer, the host-to-host transport layer, and the application layer.

In the following sections, we describe the function of each layer in more detail, starting with the network access layer and working our way up to the application layer.

Network Access Layer

The *network access layer* is the lowest layer in the Internet reference model. This layer contains the protocols that the computer uses to deliver data to the other computers and devices that are attached to the network. The protocols at this layer perform three distinct functions:

- They define how to use the network to transmit a *frame*, which is the data unit passed across the physical connection.
- They exchange data between the computer and the physical network.
- They deliver data between two devices on the same network. To deliver data on the local network, the network access layer protocols use the physical addresses of the nodes on the network. A physical address is stored in the network adapter card of a computer or other device, and it is a value that is “hardcoded” into the adapter card by the manufacturer.

Unlike higher level protocols, the network access layer protocols must understand the details of the underlying physical network, such as the packet structure, maximum frame size, and the physical address scheme that is used. Understanding the details and constraints of the physical network ensures that these protocols can format the data correctly so that it can be transmitted across the network.

Internetwork Layer

In the Internet reference model, the layer above the network access layer is called the *internetwork layer.* This layer is responsible for routing messages through internetworks. Two types of devices are responsible for routing messages between networks. The first device is called a *gateway*, which is a computer that has two network adapter cards. This computer accepts network packets from one network on one network card and routes those packets to a different network via the second network adapter card. The second device is a *router*, which is a dedicated hardware device that passes packets from one network to a different network. These two terms are often used interchangeably, but distinct differences exist in their ability to route packets and their roles within Cisco Centri Firewall.

The internetwork layer protocols provide a datagram network service. *Datagrams* are packets of information that comprise a header, data, and a trailer. The header contains information, such as the *destination address*, that the network needs to route the datagram. A header can also contain other information, such as the *source address* and *security labels*. Trailers typically contain a *checksum value*, which is used to ensure that the data is not modified in transit.

The communicating entities—which can be computers, operating systems, programs, processes, or people—that use the datagram services must specify the destination address (using control information) and the data for each message to be transmitted. The internetwork layer protocols package the message in a datagram and send it off.

A datagram service does not support any concept of a session or connection. Once a message is sent or received, the service retains no memory of the entity with which it was communicating. If such a memory is needed, the protocols in the host-to-host transport layer maintain it. The abilities to retransmit data and check it for errors are minimal or nonexistent in the datagram services. If the receiving datagram service detects a transmission error during transmission using the checksum value of the datagram, it simply ignores (or drops) the datagram without notifying the receiving higher-layer entity.

Host-to-Host Transport Layer

The protocol layer just above the internetwork layer is the *host-to-host transport layer.* It is responsible for providing end-to-end data integrity and provides a highly reliable communication service for entities that want to carry out an extended two-way conversation.

In addition to the usual transmit and receive functions, the host-to-host transport layer uses *open* and *close* commands to initiate and terminate the connection. This layer accepts information to be transmitted as a *stream* of characters, and it returns information to the recipient as a stream.

The service employs the concept of a connection (or *virtual circuit*). A *connection* is the state of the host-to-host transport layer between the time that an open command is accepted by the receiving computer and the time that the close command is issued by either computer.

Application Layer

The top layer in the Internet reference model is the *application layer.* This layer provides functions for users or their programs, and it is highly specific to the application being performed. It provides the services that user applications use to communicate over the network, and it is the layer in which user-access network processes reside. These processes include all of those that users interact with directly, as well as other processes of which the users are not aware.

This layer includes all applications protocols that use the host-to-host transport protocols to deliver data. Other functions that process user data, such as data encryption and decryption and compression and decompression, can also reside at the application layer.

The application layer also manages the sessions (connections) between cooperating applications. In the TCP/IP protocol hierarchy, sessions are not identifiable as a separate layer, and these functions are performed by the host-to-host transport layer. Instead of using the term “session,” TCP/IP uses the terms “socket” and “port” to describe the path (or virtual circuit) over which cooperating applications communicate. However, in describing Cisco Centri Firewall, we do distinguish between sessions and ports. A *session* is a connection over a TCP or UDP port that is made between two computers, either one of which is protected by Cisco Centri Firewall.

Most of the application protocols in this layer provide user services, and new user services are added often. For cooperating applications to be able to exchange data, they must agree about how data is represented. The application layer is responsible for standardizing the presentation of data.

In the following section, we provide a history of TCP/IP and then define the TCP/IP protocol suite using the Internet reference model.

What is TCP/IP?

The name TCP/IP refers to a suite of data communication protocols. The name is misleading because TCP and IP are only two of dozens of protocols that compose the suite. Its name comes from two of the more important protocols in the suite: the *Transmission Control Protocol* (TCP) and the *Internet Protocol* (IP).

TCP/IP originated out of the investigative research into networking protocols that the Department of Defense (DoD) initiated in 1969. In 1968, the DoD Advanced Research Projects Agency (ARPA) began researching the network technology that is now called *packet switching*.

The original focus of this research was to facilitate communication among the DoD community. However, the network that was initially constructed as a result of this research, then called ARPANET, gradually became known as the Internet. The TCP/IP protocols played an important role in the development of the Internet. In the early 1980s, the TCP/IP protocols were developed. In 1983, they became standard protocols for ARPANET.

Because of the history of the TCP/IP protocol suite, it is often referred to as the *DoD protocol suite* or the *Internet protocol suite*.

How TCP/IP Works

In this section, we describe some of the protocols that compose TCP/IP using the Internet reference model. We also define the function of each protocol and define terms that are specific to TCP/IP.

Network Access Layer

The design of TCP/IP hides the function of this layer from users—it is concerned with getting data across a specific type of physical network (such as Ethernet, Token Ring, etc.). This design reduces the need to rewrite higher levels of a TCP/IP stack when new physical network technologies are introduced (such as ATM and Frame Relay).

The functions performed at this level include encapsulating the IP datagrams into *frames* that are transmitted by the network. It also maps the IP addresses to the physical addresses used by the network. One of the strengths of TCP/IP is its addressing scheme, which uniquely identifies every computer on the network. This IP address must be converted into whatever address is appropriate for the physical network over which the datagram is transmitted.

Data to be transmitted is received from the internetwork layer. The network access layer is responsible for routing and must add its routing information to the data. The network access layer information is added in the form of a header, which is appended to the beginning of the data.

In Windows NT, the protocols in this layer appear as NDIS drivers and related programs. The modules that are identified with network device names usually encapsulate and deliver the data to the network, while separate programs perform related functions such as address mapping.

Internetwork Layer

The best known TCP/IP protocol at the internetwork layer is the *Internet Protocol (*IP*),* which provides the basic packet delivery service for all TCP/IP networks. In addition to the physical node addresses used at the network access layer, the IP protocol implements a system of logical host addresses called IP addresses. The IP addresses are used by the internetwork and higher layers to identify devices and to perform internetwork routing. The Address Resolution Protocol (ARP) enables IP to identify the physical address that matches a given IP address.

IP is used by all protocols in the layers above and below it to deliver data, which means all TCP/IP data flows through IP when it is sent and received, regardless of its final destination.

Internet Protocol

IP is a *connectionless protocol*, which means that IP does not exchange control information (called a *handshake*) to establish an end-to-end connection before transmitting data. In contrast, a *connection-oriented protocol* exchanges control information with the remote computer to verify that it is ready to receive data before sending it. When the handshaking is successful, the computers are said to have established a *connection*. IP relies on protocols in other layers to establish the connection if connection-oriented services are required.

IP also relies on protocols in another layer to provide error detection and error recovery. Because it contains no error detection or recovery code, IP is sometimes called an *unreliable protocol*.

The functions performed at this layer are as follows:

- **Define the datagram, which is the basic unit of transmission in the Internet.** The TCP/IP protocols were built to transmit data over the ARPANET, which was a *packet switching network*. A *packet* is a block of data that carries with it the information necessary to deliver it—in a manner similar to a postal letter that has an address written on its envelope. A packet switching network uses the addressing information in the packets to switch packets from one physical network to another, moving them toward their final destination. Each packet travels the network independently of any other packet. The *datagram* is the packet format defined by IP.
- **Define the Internet addressing scheme.** IP delivers the datagram by checking the destination address in the header. If the destination address is the address of a host on the directly attached network, the packet is delivered directly to the destination. If the destination address is not on the local network, the packet is passed to a gateway for delivery. *Gateways* and *routers* are devices that switch packets between the different physical networks. Deciding which gateway to use is called *routing*. IP makes the routing decision for each individual packet.
- **Move data between the Network Access Layer and the Host-to-Host Transport Layer.** When IP receives a datagram that is addressed to the local host, it must pass the data portion of the datagram to the correct host-to-host transport layer protocol. This selection is done by using the *protocol number* in the datagram header. Each host-to-host transport layer protocol has a unique protocol number that identifies it to IP.
- **Route datagrams to remote hosts.** Internet gateways are commonly (and perhaps more accurately) referred to as IP routers because they use IP to route packets between networks. In traditional TCP/IP jargon, there are only two types of network devices:

gateways and hosts. Gateways forward packets between networks and hosts do not. However, if a host is connected to more than one network (called a *multi-homed host*), it can forward packets between the networks. When a multi-homed host forwards packets, it acts like any other gateway and is considered to be a gateway.

- **Fragment and reassemble datagrams.** As a datagram is routed through different networks, it may be necessary for the IP module in a gateway to divide the datagram into smaller pieces. A datagram received from one network may be too large to be transmitted in a single packet on a different network. This condition only occurs when a gateway interconnects dissimilar physical networks.

  Each type of network has a *maximum transmission unit (MTU)*, which is the largest packet it can transfer. If the datagram received from one network is longer than the other network's MTU, it is necessary to divide the datagram into smaller fragments for transmission. This division process is called *fragmentation*.

Internet Control Message Protocol

The *Internet Control Message Protocol* (ICMP) is part of the internetwork layer and uses the IP datagram delivery facility to send its messages. ICMP sends messages that perform the following control, error reporting, and informational functions for the TCP/IP protocol suite:

- **Flow control**. When datagrams arrive too quickly for processing, the destination host or an intermediate gateway sends an ICMP *source quench message* back to the sender. This message instructs the source to stop sending datagrams temporarily.
- **Detect unreachable destinations.** When a destination is unreachable, the computer detecting the problem sends a *destination unreachable message* to the datagram's source. If the unreachable destination is a network or host, the message is sent by an intermediate gateway. But if the destination is an unreachable port, the destination host sends the message.
- **Redirect routes.** A gateway sends the ICMP *redirect message* to tell a host to use another gateway, presumably because the other gateway is a better choice. This message can only be used when the source host is on the same network as both gateways.
- **Check remote hosts.** A host can send the ICMP *echo message* to see if a remote computer's IP is up and operational. When a computer receives an echo message, it sends the same packet back to the source host.

Host-to-Host Transport Layer

The protocol layer just above the internetwork layer is the *host-to-host layer.* It is responsible for end-to-end data integrity. The two most important protocols employed at this layer are the *Transmission Control Protocol* (TCP) and *User Datagram Protocol* (UDP).

TCP provides reliable, full-duplex connections and reliable service by ensuring that data is resubmitted when transmission results in an error (end-to-end error detection and correction). Also, TCP enables hosts to maintain multiple, simultaneous connections. When error correction is not required, UDP provides unreliable datagram service (connectionless) that enhances network throughput at the host-to-host transport layer.

Both protocols deliver data between the *application layer* and the *internetwork layer.* Applications programmers can choose the service that is most appropriate for their specific applications.

User Datagram Protocol

The *User Datagram Protocol* gives application programs direct access to a datagram delivery service, like the delivery service that IP provides. This direct access allows applications to exchange messages over the network with a minimum of protocol overhead.

UDP is an unreliable, connectionless datagram protocol. “Unreliable” merely means that the protocol has no technique for verifying that the data reached the other end of the network correctly. Within your computer, UDP will deliver data correctly.

Why do applications programmers choose UDP as a data transport service? A number of good reasons exist. If the amount of data being transmitted is small, the overhead of creating connections and ensuring reliable delivery may be greater than the work of retransmitting the entire data set. In this case, UDP is the most efficient choice for a host-to-host transport layer protocol.

Applications that fit a “query-response” model are also excellent candidates for using UDP. The response can be used as a positive acknowledgment to the query. If a response is not received within a certain time period, the application just sends another query. Still other applications provide their own techniques for reliable data delivery and do not require that service from the transport layer protocol. Imposing another layer of acknowledgment on any of these types of applications is redundant.

Transmission Control Protocol

Applications that require the host-to-host transport protocol to provide reliable data delivery use TCP because it verifies that data is delivered across the network accurately and in the proper sequence. TCP is a *reliable, connection-oriented, byte-stream* protocol.

Application Layer

The most widely known and implemented TCP/IP application layer protocols are listed below:

- **File Transfer Protocol (FTP).** Performs basic interactive file transfers between hosts.
- **Telnet.** Enables users to execute terminal sessions with remote hosts.
- **Simple Mail Transfer Protocol (SMTP).** Supports basic message delivery services.
- **HyperText Transfer Protocol (HTTP).** Supports the low-overhead transport of files consisting of a mixture of text and graphics. It uses a stateless, connection- and object-oriented protocol with simple commands that support selection and transport of objects between the client and the server.

In addition to widely known protocols, the application layer includes the following protocols:

- **Domain Name Service (DNS).** Also called *name service*; this application maps IP addresses to the names assigned to network devices.
- **Routing Information Protocol (RIP).** Routing is central to the way TCP/IP works. RIP is used by network devices to exchange routing information.
- **Simple Network Management Protocol (SNMP).** A protocol that is used to collect management information from network devices.
- **Network File System (NFS).** A system developed by Sun Microsystems that enables computers to mount drives on remote hosts and operate them as if they were local drives.

Some protocols, such as Telnet and FTP, can only be used if the user has some knowledge of the network. Other protocols, like RIP, run without the user even knowing that they exist.

Throughout this guide, we describe how Cisco Centri Firewall allows you to control access to these application layer protocols. This basic description of TCP/IP and the Internet reference model given in this appendix provides a basis for understanding what it is that Cisco Centri Firewall does and how it does it.